Turbo Codes for Wireless Mobile Communication Systems Applications

Dissertation

Moataz Mohamed Salah, Major, Egypt

AFIT/DS/ENG/00-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the U. S. Department of Defense, the U. S. Government, the Egyptian Ministry of Defense, or the Egyptian Government.

AFIT/DS/ENG/00-01

Turbo Codes for Wireless Mobile Communication Systems Applications

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Moataz Mohamed Salah, B.S., M.S.

Major, Egypt

June 2000

AFIT/DS/ENG/00-01

Turbo Codes for Wireless Mobile Communication Systems Applications

Moataz Mohamed Salah, B.S., M.S.
Major, Egypt

Approved:

	Date
Richard A. Raines (Chairman)	2 Jun 00
William F. Bailey (Dean's Representative)	31 MAY 00
Michael A. Temple (Member)	31 May 00
Thomas G. Bailey (Member)	31 May 00

Accepted:

Robert A. Calico, Jr. — 8 Jun 00
Dean, Graduate School of Engineering and Management

Acknowledgments

At first, I thank God for giving me the ability to finish this research effort and to learn a little more about one small aspect of his universe. I would like also to thank the Egyptian government for giving me the opportunity to study at the Air Force Institute of Technology.

I would like to thank my advisor, Major Richard A. Raines, Associate Professor of Electrical Engineering. His frequent assistance, encouragement and suggestions allowed me to make steady progress throughout the research phase. I extend my gratitude to my committee members, Dr. Michael A. Temple and Dr. Thomas G. Bailey, for their assistance, cooperation and comments.

I extend my special thanks to all the members of the library staff. Without their support this work would be sorely lacking. I also want to thank David Doak, system administrator, for his great assistance.

I would like to thank Mrs. Annette Robb, Director of International Affairs, and all the personnel in the IMSO for their assistance and help. I would like also to thank Mr. Vic Grazier who has helped me and the other international students and our families since our arrival here.

Finally and most importantly, I want to thank my wife Hala, my daughter Laila and my son Khalid for their patience and continuous support in overcoming the difficulties of these years.

Moataz M. Salah

Table of Contents

List of Figures

List of Tables

Abstract

Research in coding theory has seen many proposals aimed at the construction of powerful codes using block and convolutional codes. Recently, a new forward error control code, known as turbo code, was introduced. This new code yields very good performance (near the Shannon limit) with relatively simple component codes and large interleavers in combination with an iterative decoding process.

Among the most important applications of turbo codes are wireless mobile communication systems. A significant performance metric for this application is the minimization of end-to-end delay. In this dissertation, different ways to enhance the performance of turbo codes with short frames are presented. One way of enhancing the performance of turbo codes with short frames is by optimizing the energy allocation strategies to the output bitstreams. For turbo codes with short frames, different ways to allocate the energy are investigated using computer simulation and modified analytic bounds. The results show that the performance is improved without any increase in complexity. Another way to enhance the performance is with the proper design of the interleaver. This work proposes a new and unique interleaver for equal and unequal error protections. In equal error protection applications, the proposed interleaver performance outperforms the other conventional interleavers at higher signal-to-noise ratios. The introduction of this interleaver allows for less complex hardware implementations as the single interleaver replaces multiple interleavers performing the task.

A novel analytical bound is developed to evaluate the performance of punctured turbo codes, along with its applications to different channel types. The new approach introduces a random device, the *hypergeometric-puncturing device*, which

averages the output weight of the punctured codeword over all the punctured positions. Before the development of this novel bound, the only way to study the performance of punctured turbo codes was via computer simulation. Introduction of this bound allows the analytical study to be extended to higher signal-to-noise ratios where the computer simulation is not effective. This bound serves to illustrate the achievable performance of turbo codes and the effects of block length and constituent encoder choice in the performance of turbo codes.

Turbo Codes for Wireless Mobile Communication Systems Applications

1 Introduction

1.1 Error control Coding

In recent years, wireless mobile communication systems have grown very rapidly to transmit voice, image, and data. Wireless mobile communication systems present several design challenges resulting from the mobility of users throughout the system and the time-varying channel. Also, there has been an increasing demand for efficient and reliable digital communication systems. The major concern of design engineers is to minimize the received error probability by making efficient use of power and bandwidth resources, while keeping system complexity reasonable to reduce cost.

Figure 1-1 shows the basic configuration of a point-to-point digital communication link. The data to be transmitted over this link can come from an analog source, in which case it is first converted into digital format, or a direct digital information source. If the data represents a speech signal, the digitizer is implemented as a speech codec. Usually, a digital data is source encoded to remove unnecessary redundancy from the data (data compression). The channel encoder operates on the compressed data and introduces controlled redundancy prior to transmission over the channel. The modulator converts the discrete channel symbols into waveforms which are transmitted through the channel. Upon reception, the demodulator reconverts the waveforms into a discrete sequence of symbols. The decoder then reproduces an estimate of the compressed input data sequence which is subsequently reconverted into the original signal or data sequence.

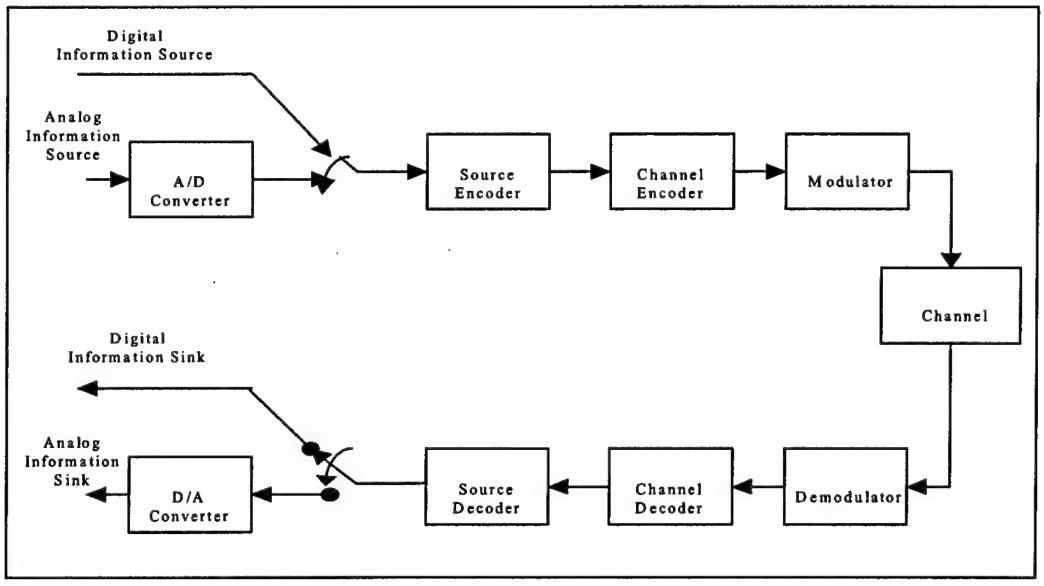

Figure 1-1 Block diagram of a point-to-point digital communication link.

In 1948, Shannon demonstrated [1] that by proper information encoding, errors induced by a noisy channel could be reduced to any desired level without sacrificing information transfer rate. The most famous formula from Shannon's work is the channel capacity of an ideal band-limited Gaussian channel which is given by:

$$C = W \log_2(1 + S/N) \qquad \text{bits/s} \qquad (1\text{-}1)$$

where C is the channel capacity, i.e., the maximum number of bits that can be transmitted through the channel per unit of time, W is the bandwidth of the channel in Hz, and S/N is the signal-to-noise power ratio at the receiver. Shannon's theorem asserts that error probabilities as small as desired can be achieved as long as the transmission rate R_s through the channel is smaller than the channel capacity, C. This can be achieved using an appropriate encoding and decoding operation. For information rates $R_s > C$, it is not possible to find a code that can achieve an arbitrary small error probability.

Shannon did not indicate how this could be achieved. Subsequent research has led to a number of techniques that introduce structured redundancy to allow for error correction without retransmission. These techniques, collectively known as forward error correction (FEC) coding techniques or channel coding techniques, are used in systems where a reverse channel is not available for requesting retransmission of incorrect frames.

Since Shannon's work, many efforts have investigated the problem of devising efficient encoding and decoding. Coding theorists have traditionally attacked the problem of designing good codes by developing codes with structure that leads to feasible decoders. Coding theory suggests that codes chosen "at random" should perform well if the block size is large enough. The challenge of finding practical decoders for

almost random large codes has not been seriously considered until recently. Perhaps the most exciting and potentially important development in coding theory in recent years has been the dramatic announcement of "Turbo Codes" invented by French researchers in 1993 [2]. Turbo codes are a parallel concatenation of two constituent convolutional codes separated by an interleaver and the decoder working in an iterative fashion. This new coding technique has been considered as a candidate for many applications, including deep space communications and wireless mobile communication systems, due to its very good performance.

1.2 Research Goals

In 1993 a new error correcting technique, known as turbo coding, was introduced [2] and claimed to achieve near Shannon-limit error correction performance; a required E_b/N_0 of 0.7 db was reported for bit error rates (BER) of 10^{-5} using a code rate of 1/2 in an Additive White Gaussian Noise (AWGN) channel.

Turbo code techniques are based on encoding on a frame-by-frame basis (size of the interleaver). The interleaver size greatly determines the performance of turbo codes.

The goal of this research is two-fold. The first is to study the turbo code performance and the encoder and decoder structures used to generate the codes. The research goal is to examine ways to enhance turbo code performance in a mobile wireless environment. To successfully use turbo codes in wireless mobile communication systems for speech transmission, the following two requirements must be met. First, the interleaver must be designed in such a way that the maximum delay in speech

transmission is not exceeded. Second, the turbo decoder signal processing delay must not be large.

The second goal of the research is to extend the body of knowledge related to the mathematical analysis of bit error probability for punctured turbo codes. This will facilitate studying the performance of punctured turbo codes at higher signal-to-noise ratios.

1.3 Dissertation Outline

The dissertation is organized in the following manner. Chapter 2 presents the characteristics and parameters of the wireless communication channel. The traditional channel coding techniques of block, convolutional, and concatenated codes are presented. An extensive review of literature covering turbo codes, the details of choosing its individual components, its performance, and its applications to different wireless communication systems are also covered.

In Chapter 3, the principle of the iterative decoding process of the turbo decoder is presented. Turbo decoders consist of two soft-input/soft-output constituent convolutional decoders working together in an iterative fashion. Both constituent decoders accept soft input and deliver soft output. The problem of estimating the state sequence of a Markov process observed through noise using the trellis-based decoding algorithms is addressed. The Viterbi Algorithm and its modified version, the Soft Output Viterbi Algorithm (SOVA), are presented.

The research methodologies applied throughout this thesis were mathematical analysis and computer simulations. In Chapter 4, the methodology used to develop the mathematical models of an analytical bound and its calculations are presented. Also, this chapter covers simulation model development for the proposed turbo code. An algorithm to build a simulation model of the channel under consideration is also presented.

Chapter 5 addresses the problem of enhancing turbo code performance using short frames. One way of enhancing performance is to optimize the energy allocated to each bitstream (turbo codes have three output bitstreams: systematic, first encoder, and second encoder) to achieve the best possible performance. In standard turbo codes, all bits are transmitted with equal energy. Changing the energy allocation strategy can enhance turbo code performance using short frame lengths.

Another way to enhance performance is in the interleaver design. With proper design, the pairing of low-weight sequences into the encoders can be avoided. Circular shift interleavers can be used to ensure that the minimum distance due to weight-2 input sequences grows roughly as $\sqrt{2N}$, where N is the block length. A generalization of the circular shift interleaver mapping function for both equal and unequal error protections is presented.

Chapter 6 presents background discussion of punctured convolutional codes and useful definitions and notations as used in the derivation of the analytical punctured bound. The punctured bound derivation is presented. Applications of the punctured bound to AWGN, fully-interleaved fading, and correlated fading channels are presented.

Performance evaluation of these bounds and a comparison of the analytical bound with simulation results are also presented.

Chapter 7 concludes the dissertation with a summary of the work presented. Recommendations for research extensions are also discussed.

2 Background

2.1 Introduction

In 1948, Shannon [1] demonstrated in a landmark paper that, by proper encoding of the information, errors induced can be reduced to any desired level without sacrificing the rate of information transfer. The Shannon limit of -1.59 dB is the minimum amount of signal-to-noise ratio (E_b/N_0) that is necessary to achieve an arbitrarily low bit error probability over an Additive White Gaussian Noise (AWGN) channel. Since Shannon's work, numerous research efforts have investigated the problem of devising efficient encoding and decoding. Perhaps the most exciting and potentially important development in coding theory in recent years has been the dramatic announcement of "Turbo codes" [2]. Turbo code performance is claimed to achieve near Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required E_b/N_0 of 0.7 dB was reported for a Bit Error Rate (BER) of 10^{-5} and code rate of 1/2.

Turbo codes are constructed by applying two or more component codes to different interleaved versions of the same information sequence. Then, the encoded bits are decoded through an iterative decoding algorithm of relatively low complexity.

Since the first publication regarding turbo codes in 1993 [2], the following aspects of turbo codes have been under research: (i) optimization of the constitute encoder and decoder structures, and (ii) optimization of the interleaving function. Applications of turbo codes to different wireless communication channels and to speech transmission (short frame applications) are also under research.

Turbo codes are based on the encoding on a frame-by-frame basis (size of the interleaver). The size of the interleaver greatly determines the performance (BER) of the turbo code. In the foundational work by [2], a frame size of 65,532 information bits was used to achieve a BER in an AWGN channel of 10^{-5} at E_b/N_0 of 0.7 dB. By comparison, the speech frames contain less than 200 bits to be processed by the channel encoder. In the case of Global System for Mobile communications (GSM) [3], the speech frames contain 189 bits. Also in the case of DS-CDMA digital cellular system, known as Interim Standard (IS-95), the speech frames contain 192 bits [4].

In this chapter we address the problem of optimizing the components of turbo code and their applications to speech transmission systems. There are different proposed configurations of turbo codes-two parallel concatenated codes [2], multiple parallel concatenated codes [5], serial concatenated codes [6-8], and hybrid concatenated codes [8, 9]. In our survey we concentrate on the original turbo code proposed in [2], which uses the parallel concatenation.

Section 2.2 reviews the characteristics and parameters of the wireless communication channel. In Section 2.3, the traditional channel coding techniques-block, convolutional, and concatenated codes are presented. Section 2.4 presents an intensive

review of the literature that covers the turbo code. The details of choosing the individual components of turbo codes, its performance, and its applications to different wireless communication systems are also covered in Section 2.4. Section 2.5 summarizes the contents of the chapter.

2.2 The Wireless Mobile Channel

In the study of communication systems, the classical (ideal) Additive White Gaussian Noise (AWGN) channel is the usual starting point for understanding basic performance relationships. The additive noise introduced by hardware components is at the front end receiver. The channel model is illustrated in Figure 2-1, where the noise term $n(t)$ is a stochastic process. Receiver noise is commonly modeled as a zero-mean Gaussian process. When the noise is white, the channel is referred to as an Additive White Gaussian Noise (AWGN) channel.

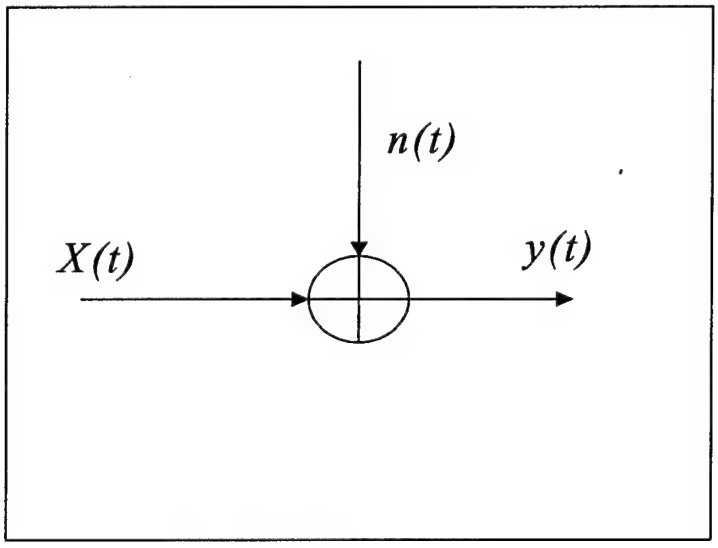

Figure 2-1 Additive noise channel.

The transmission path between the transmitter and the receiver in a wireless mobile communication system can vary from simple line-of-sight to one that is severely obstructed by buildings, mountains, and foliage. Even the speed of motion impacts how rapidly the signal level fades. A signal can travel from transmitter to receiver over multiple reflective paths. This phenomenon is referred to as multipath propagation. The effect can cause fluctuations in the received signal's amplitude, phase, and angle of arrival giving rise to the terminology multipath fading [10]. The end-to-end modeling and design of systems that mitigate the effects of fading are usually more challenging than those whose sole source of performance degradation is AWGN. The wireless mobile channel places fundamental limitations on the performance (E_b/N_0 and BER) of a wireless mobile communication system. Modeling the radio channel has historically been one of the most difficult parts of mobile radio system design, and is typically performed in a statistical fashion, based on measurements made specifically for an intended communication system or spectrum allocation [11, 12].

In order for systems engineers to be able to determine optimum methods of mitigating the impairments caused by the channel, it is essential that the transmission channel be accurately modeled. It seems reasonable, therefore, to consider mobile radio channels as special cases of random time-invariant linear filters. Two types of fading effects characterize mobile communication: large-scale and small-scale fading. Large scale fading represents the average signal power attenuation or path loss due to motion over large areas [10, 13]. This phenomenon is affected by prominent terrain contours (hills, forests, clumps of buildings, etc.) between the transmitter and the receiver. The receiver is often represented as being "shadowed" by such prominence. The statistics of

large-scale fading provide a way of computing an estimate of path loss as a function of distance. Small-scale fading refers to the dramatic changes in signal amplitude over a short period of time or travel distance.

As the mobile moves over very small distances, the instantaneous received signal strength may fluctuate rapidly giving rise to small-scale fading. The reason for this is that the received signal is a sum of many contributions coming from different directions [14]. In small-scale fading, the received signal power may vary by as much as three or four orders of magnitude (30 or 40 db) when the receiver is moved by only a fraction of a wavelength.

Small-scale fading, or simply *fading*, is used to describe the rapid fluctuations of the amplitude of a radio signal over a short period of time or travel distance, so that the large scale path loss effects may be ignored. Fading is caused by interference between two or more versions of the transmitted signal, which arrive at the receiver at slightly different times. These waves, called *multipath waves*, combine at the receiver antenna to give a resultant signal which can vary widely in amplitude and phase. When the received signal is made up of multiple reflective rays plus a significant line-of-sight (non faded) component, the envelope amplitude due to small-scale fading has a Rician probability density function (pdf), and is referred to as Rician fading [14]. The nonfaded component is called the specular component. As the amplitude of the specular component approaches zero, the Rician pdf approaches a Rayleigh pdf, and is expressed as follows:

$$P_{Rayleigh}(r) = \frac{r}{\sigma^2} e^{\frac{r^2}{2\sigma^2}}, \qquad r \geq 0 \tag{2-1}$$

where r is the envelope amplitude of the received signal and σ^2 is the variance of the received signal r. The Rayleigh pdf results from having no specular component of the signal, thus for a single link it represents the pdf associated with the worst case of fading.

2.2.1 The Mobile Multipath Channel Parameters

In order to compare different multipath channels and to develop some general design guidelines for wireless systems, parameters which grossly quantify the multipath channel are used.

- *Maximum excess delay*: For a typical wireless radio channel, the received signal usually consists of several discrete multipath components. For some channels, such as the tropospheric scatter channel, received signals are often seen as a continuum of multipath components [10]. For a single transmitted impulse, the time, T_m, between the first and last received component represents the maximum excess delay, during which the multipath signal power falls to some threshold level below that of the strongest component. The threshold level might be chosen at 10 or 20 db below the level of the strongest component. Note that for the ideal case, the excess delay would be zero.

- *Root mean square delay spread*: The maximum excess delay, T_m, is not necessarily the best indicator of how any given system will perform on a channel because different channels with the same value of T_m can exhibit very different profiles of signal intensity, $S(\tau)$, over the delay span. Knowledge of a multipath intensity profile, $S(\tau)$, helps answer the question "For a transmitted impulse, how does the average received power vary as a function of time delay, τ ?" The term "time delay" is used to refer to the

excess delay. It represents the signal's propagation delay that exceeds the delay of the first signal arrival at the receiver. A more useful measurement of delay spread is most often characterized in terms of the root mean square (rms) delay spread, σ_τ, where:

$$\sigma_\tau = \sqrt{\overline{\tau^2} - (\bar{\tau})^2} \tag{2-2}$$

and $\bar{\tau}$ is the mean excess delay, $\overline{\tau^2}$ is the square root of the second central moment of $S(\tau)$ [14].

- *Coherence bandwidth*: The coherence bandwidth, f_0, is a statistical measure of the range of frequencies over which the channel passes all spectral components with approximately equal gain and linear phase (channel can be considered flat over this range of frequencies). An exact relationship between coherence bandwidth and delay spread does not exist. Several approximated relationships have been described in [10, 14].

- *Doppler spread and Coherence time*: Delay spread and coherence bandwidth are parameters which describe the time dispersive nature of the channel in a local area. However, they do not offer information about the time varying nature of the channel caused by either relative motion between the mobile and base station, or by movement of objects in the channel. Doppler spread, f_d, and coherence time, T_c, are parameters which describe the time varying nature of the channel in a small-scale region.

The following example illustrates the Doppler phenomena [14]. Consider a mobile moving at a constant velocity, v, along a path segment having length d between points X and Y, while it receives signals from a remote source, S. This is illustrated in Figure 2-2.

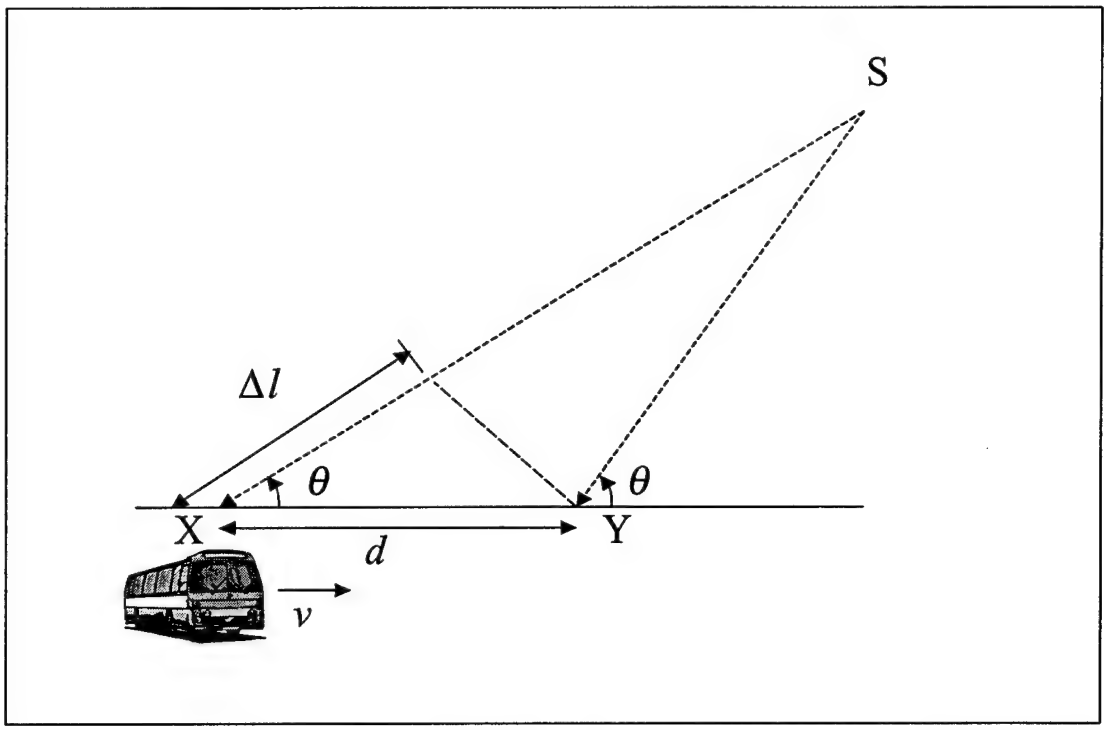

Figure 2-2 Illustration of Doppler shift [14].

The difference in path lengths traveled by the wave from source, *S*, to the mobile points *X* and *Y* is $\Delta l = d\cos\theta = v\,\Delta t\cos\theta$, where Δt is the time required for the mobile to travel from *X* to *Y*, and θ is assumed to be the same at points *X* and *Y* since the source is assumed to be very far away. The phase change in the received signal due to the difference in path lengths is therefore:

$$\Delta\phi = \frac{2\pi\Delta l}{\lambda} = \frac{2\pi v\Delta t}{\lambda}\cos\theta \tag{2-3}$$

And hence, the apparent change in frequency or Doppler shift, B_d, is:

$$B_d = \frac{1}{2\pi}\frac{\Delta\phi}{\Delta t} = \frac{v}{\lambda}\cos\theta \tag{2-4}$$

Equation 2-4 relates the Doppler shift to the mobile velocity, the spatial angle between the direction of the motion of the mobile and the direction of the arrival of the waves, and the wave length of the carrier signal (carrier frequency).

In a typical multipath environment, the received signal arrives from several reflected paths with different path distances and different angles of arrivals, and the Doppler shift of each arriving path is generally different from that of another path. The effect on the received signal is seen as a Doppler spreading or spectral broadening of the transmitted signal frequency, rather than a shift. The Doppler spread, f_d, is sometimes called the fading bandwidth, and it is defined as the maximum Doppler shift as follows:

$$f_d = \frac{v}{\lambda} \tag{2-5}$$

If the baseband signal bandwidth is much greater than the Doppler spread, f_d, the effects of Doppler spread is negligible at the receiver [14].

Coherence time, T_c, is the expected time duration over which the channel's response to a sinusoid is essentially invariant. When T_c is defined more precisely as the time duration over which the channel's response to a sinusoid has a correlation greater than 0.5, the relationship between T_c and f_d is approximately [10] as follows:

$$T_c \approx \frac{9}{16\pi f_d} \tag{2-6}$$

Coherence time, T_c, is used to characterize the time varying nature of the frequency dispersiveness of the channel in the time domain.

2.2.2 Multipath Fading Channel Types

Depending on the relation between the signal parameters (such as bandwidth, symbol period, etc.) and the channel parameters (such as rms delay spread and Doppler spread), different transmitted signals will undergo different types of fading. The time dispersion and frequency dispersion mechanisms in a mobile radio channel lead to four possible distinct effects [10, 14]; flat fading, frequency selective fading, fast fading, and slow fading. The two propagation mechanisms, multi path delay and Doppler spread, are independent of one another.

Table 2-1 shows the four different types of fading, the conditions to occur, the explanation of the effect of this type of fading to the reception of the symbol, the associated degradation, and the mitigation techniques for combatting the degradation. In Table 2-1, we denote the bandwidth of the signal by B_s, the symbol by T_s, the maximum delay spread by T_m, the coherence bandwidth of the channel by f_0, the intersymbol interference by ISI, and signal-to-noise ratio by SNR.

From Table 2-1, summarizing the conditions that must be met so that the channel does not introduce frequency selective and fast fading distortion, we need Equation 2-7 or 2-8 to be satisfied.

$$f_0 > B_s > f_d \tag{2-7}$$

$$T_m < T_s < T_c \tag{2-8}$$

We want the channel coherence bandwidth to exceed the signaling rate, which in turn should exceed the fading rate of the channel [15].

Table 2-1 Types of multipath fading, conditions, effects, degradations, and mitigations.

Fading type	Fading based on time dispersion due to multipath		Fading based on Doppler spread	
	Flat fading	Frequency selective fading	Fast fading	Slow fading
Conditions to occur	$T_m < T_s$ $B_s < f_0$	$T_m > T_s$ $B_s > f_0$	$T_c < T_s$ $f_d > B_s$	$T_c > T_s$ $f_d < B_s$
Effects	The received multipath components of the symbol arrive within the symbol duration	The received multipath components of the symbol extend beyond the symbol duration	The fading characteristics of the channel change several times while a symbol is propagating	The time duration that the channel behaves in a correlated manner is long compared to T_s
Degradation effect	Loss in SNR	ISI distortion	PLL failure	Loss in SNR
Mitigation	* Diversity techniques * Error control coding	* Adaptive equalization * Spread spectrum	* Robust modulation * Signal redundancy to increase signal rate * Coding and interleaving	* Diversity techniques * Error control coding

2.3 Channel Coding Techniques

In 1948, Claude Shannon issued a challenge to communications engineers by proving that the communication systems could be made arbitrarily reliable as long as a fixed percentage of the transmitted signal was redundant [1]. He did not indicate how this could be achieved. Subsequent research has led to a number of techniques that introduce redundancy to allow for correction of errors without retransmission.

Channel coding protects digital data from errors by selectively introducing redundancies in the transmitted data. Coding involves adding extra bits to the data stream so that the decoder can reduce or correct errors at the output of the receiver. However, these extra bits have the disadvantage of increasing the data rate (bits/s) and, consequently, increasing the bandwidth of the encoded signal.

Before discussing any codes, several definitions are needed. The Hamming weight of a code word is the number of binary 1 bits the word contains. The Hamming distance between two code words, denoted by d, is the number of bit positions that differ between the two words. The minimum Hamming distance of the code, d_{min}, is the minimum Hamming distance between any two code words in the code. The weight distribution or distance spectrum of a code is the number of code words for each possible weight.

Channel coding techniques can be divided into four categories [11, 12, 16, 17]; block codes, convolutional codes, concatenated codes, and turbo codes. In the next subsections, we review the first three types of codes with turbo codes being introduced in the following section.

2.3.1 Block Codes

Block codes enable a limited number of errors to be detected and corrected without retransmission. A block code is a mapping of k input binary symbols into n output binary symbols. Consequently, the block coder is a memoryless device (each bit in the code word is independent of the previous bits). Because $n > k$, the code can be selected to provide redundancy, such as parity bits, which are used by the decoder to provide some error detection and correction. The codes are denoted by (n, k), where the code rate R is defined by $R = k/n$. Practical values of R range from 1/4 to 7/8, and k ranges from 3 to several hundred. The ability of a block code to correct errors is a function of the code distance. Properties of block codes are as follows [11, 12]:

Linearity: The code is said to be linear if and only if the addition of any two codewords of the code is also a codeword. A linear code must contain the all-zero codeword.

Systematic: A systematic code is one in which the parity bits are appended to the end of the information bits.

Cyclic: Cyclic codes are block codes, such that another codeword can be obtained by taking any one codeword, shifting the bits to the right, and placing the dropped-off bits on the left.

Encoding and decoding techniques make use of the mathematical constructs known as finite fields. Finite fields are algebraic systems which contain a finite set of elements. Addition, subtraction, multiplication, and division of finite field elements is accomplished without leaving the set. Addition and multiplication must satisfy the

commutative, associative, and distributive laws [18]. For any prime number p, there exists a finite field which contains p elements. This prime field is denoted as $GF(p)$ because finite fields are also called Glois fields, in honor of their discoverer. It is also possible to extend the prime field $GF(p)$ to a field of p^m elements which is called an extension field of $GF(p^m)$, where m is a positive integer. Codes with symbols from the binary field $GF(2)$ or its extension field $GF(2^m)$ are most commonly used in digital data transmission systems, since information in these systems is always encoded in binary form in practice.

In binary arithmetic, modulo-2 addition and multiplication are used. This arithmetic is actually equivalent to ordinary arithmetic except that 2 is considered equal to 0.

There are a number of block codes that are frequently encountered in practice like Hamming codes, Harmard codes, Golay codes, cyclic codes, BCH codes, and Reed-Solomon codes. Here, we will describe four of them.

- *Hamming codes*: Hamming codes were among the first of nontrivial error correction codes [19]. There are both binary and non-binary codes. A binary hamming code has the property that:

$$(n, k) = (2^m\text{-}1, 2^m\text{-}1\text{-}m) \tag{2-9}$$

where k is the number of information bits used to form an n bit codeword, and $m \geq 2$. The number of parity symbols are $n\text{-}k = m$. For example, if $m = 3$, we have a (7,4) code. These binary codes have a minimum distance of 3, and they are capable of correcting all

single error or detecting all combinations of two or fewer errors within the block [11]. The weight distribution for the class of Hamming (n, k) codes is known and it is given as follows [12]:

$$A(Z) = \sum_{i=0}^{n} A_i Z^i$$

$$= \frac{1}{n+1}\left[(1+Z)^n + n(1+Z)^{(n-1)/2}(1-Z)^{(n+1)/2}\right] \qquad (2\text{-}10)$$

where A_i is the number of codewords of weight i.

- *Golay codes*: Golay codes are linear binary (23, 12) codes with minimum distance of 7. The extended Golay code (24, 12) is obtained by adding an overall parity to the (23, 12) code. A (24, 12) code has minimum distance of 8. The weight distributions of the Golay code (23, 12) and the extended Golay (24, 12) code are known [12]. Extended Golay codes are considerably more powerful than the Hamming codes. The extended Golay code is guaranteed to correct all triple errors.

- *Bose-Chaudhuri-Hocquenghem (BCH) codes*: BCH codes are among the most important block codes since they exist for a wide range of rates, achieve significant coding gains, and can be implemented even at high speeds [12]. BCH codes are a generalization of Hamming codes that allow multiple error correction. The block length of the code is $n = 2^m-1$ for $m \geq 3$, and the number of errors that they can correct is bounded by $t < \frac{(2^m-1)}{2}$. The most important and common class of non-binary BCH

codes is the family of codes known as Reed-Solomon codes. The (63, 47) Reed-Solomon code in U.S. Cellular Digital Packet Data (CDPD) uses $m = 6$ bits per code symbol [14].

- *Reed-Solomon codes*: One special class of the BCH codes is the non-binary set called Reed-Solomon codes. Reed-Solomon codes achieve the largest possible code minimum distance for any linear block code with the same encoder input and output block lengths. Reed-Solomon codes are capable of correcting errors which appear in bursts and are commonly used in concatenated coding systems. The minimum distance of the code is given by [11]:

$$d_{min} = n - k + 1 \tag{2-11}$$

where k is the number of data symbols being encoded, and n is the total number of code symbols in the encoded block. The code is capable of correcting any combination of t or fewer symbol errors, and t is given as following:

$$t = \frac{d_{\min} - 1}{2} = \frac{n-k}{2} \tag{2-12}$$

Thus the number of parity symbols that must be used to correct t errors is $n - k = 2t$.

2.3.2 Convolutional Codes

Convolutional codes are powerful coding schemes for wireless mobile communication systems [20]. Convolutional codes are fundamentally different from block codes in that information sequences are not grouped into distinct blocks and encoded. Instead, a continuous sequence of information bits is mapped into a continuous

sequence of encoder output bits. It can be argued that convolutional coding can achieve a larger coding gain than a block coding with the same complexity.

Figure 2-3 shows the configuration of the convolutional encoder with a coding rate of ½ with constraint length (K) of 3. The convolutional encoder consists of (K-1 = 2) shift registers and two modulo-2 adders connected to some of the shift registers. The input data sequence is fed to the shift registers, and two output data are calculated using the contents of the shift registers. The generator polynomials determine the encoding process. For the convolutional encoder shown in Figure 2-3, two coded bits are produced when a single source bit is input.

Generally, we can define the convolutional code rate as $R = k/n$, where n is the number of coded symbols produced when k information bits are shifted to the convolutional encoder. The convolutional encoder shown in Figure 2-3 has $k = 1$ and $n = 2$.

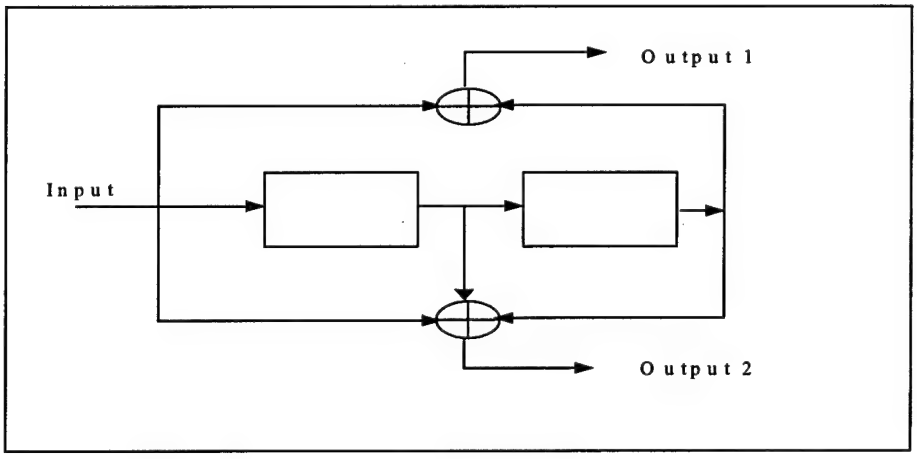

Figure 2-3 Configuration of the convolutional encoder with 1/2 code rate and K =.3.

There are various ways of representing convolutional codes such as generator polynomials, state diagrams, tree diagrams, and trellis diagrams [12].

- *Generator polynomials*: for rate 1/2, the generator polynomial can be expressed as follows:

$$G_1(x) = g_{k-1}^1 x^{k-1} + g_{k-2}^1 x^{k-2} + \ldots + g_2^1 x^2 + g_1^1 x + g_0^1 \tag{2-13}$$

$$G_2(x) = g_{k-1}^2 x^{k-1} + g_{k-2}^2 x^{k-2} + \ldots + g_2^2 x^2 + g_1^2 x + g_0^2 \tag{2-14}$$

where x indicates a 1-bit timing delay, $G_1(x)$ is the generator polynomial for the first encoded bit and $G_2(x)$ is for the second bit. The g_m^n parameters indicate whether or not an m-bit delayed source bit is added on the modulo-2 basis to obtain the *nth* encoded bit. For the encoder shown in Figure 2-3, the generator polynomials are given by:

$$G_1(x) = x^2 + 1 \tag{2-15}$$

$$G_2(x) = x^2 + x + 1 \tag{2-16}$$

The polynomial generators of a convolutional code are usually selected based on the code's free distance properties. The first criterion is to select a code that does not have catastrophic error propagation and that has the maximum free distance for the given rate and constraint length, then the number of paths at the free distance, d_{free}. The selection procedure can be further refined by considering the number of paths at $d_{free}+1$, at $d_{free}+2$, and so on, until only one code or class of codes remains. A list of best known codes of rate 1/2, $K = 3$ to 9, and rate 1/3, $K = 3$ to 8, was compiled in [21] based on this criterion.

- *State diagram*: The state diagram is simply a graph of the possible states of the encoder and the possible transitions from one state to another. The state of a convolutional encoder is determined by the contents of the right most (K-1) shift registers.

- *Tree diagram*: The tree diagram shows the structure of the encoder in the form of a tree with the branches representing the various states and the outputs of the encoder.

- *Trellis diagram*: Close observation of the tree reveals that the structure repeats itself once the number of stages is greater than the constraint length. It is observed that all branches emanating from two nodes having the same state are identical in the sense that they generate identical output sequences. This means that the two nodes having the same label can be merged. By doing this throughout the tree diagram, we can obtain another diagram called a trellis diagram which is a more compact representation.

Unlike block codes, the codewords of a convolutional code do not have fixed lengths. However, it is desirable to force the convolutional code word to be limited to a specific length. This can be done by a process called trellis termination. The trellis can be forced back to the all-zero state by setting the last K-1 bits of the input message to zeros.

There are a number of techniques for decoding convolutional codes [12]: Viterbi algorithm, Fano's sequential decoding, the stack algorithm, and feedback decoding. The most important of these methods is the Viterbi algorithm which performs maximum likelihood decoding of convolutional codes.

2.3.3 Concatenated Codes

A concatenated code is one that uses two levels of coding, an inner code and an outer code, to achieve the desired error performance. A simple concatenated code can be formed as shown in Figure 2-4. Figure 2-4 illustrates the order of encoding and decoding. The inner code, the one that interfaces with modulator/demodulator and channel, is usually configured to correct most of the channel errors. To spread any error bursts that may appear at the output of the inner coding operation, an interleaver should be inserted between the two coding steps as shown in Figure 2-4.

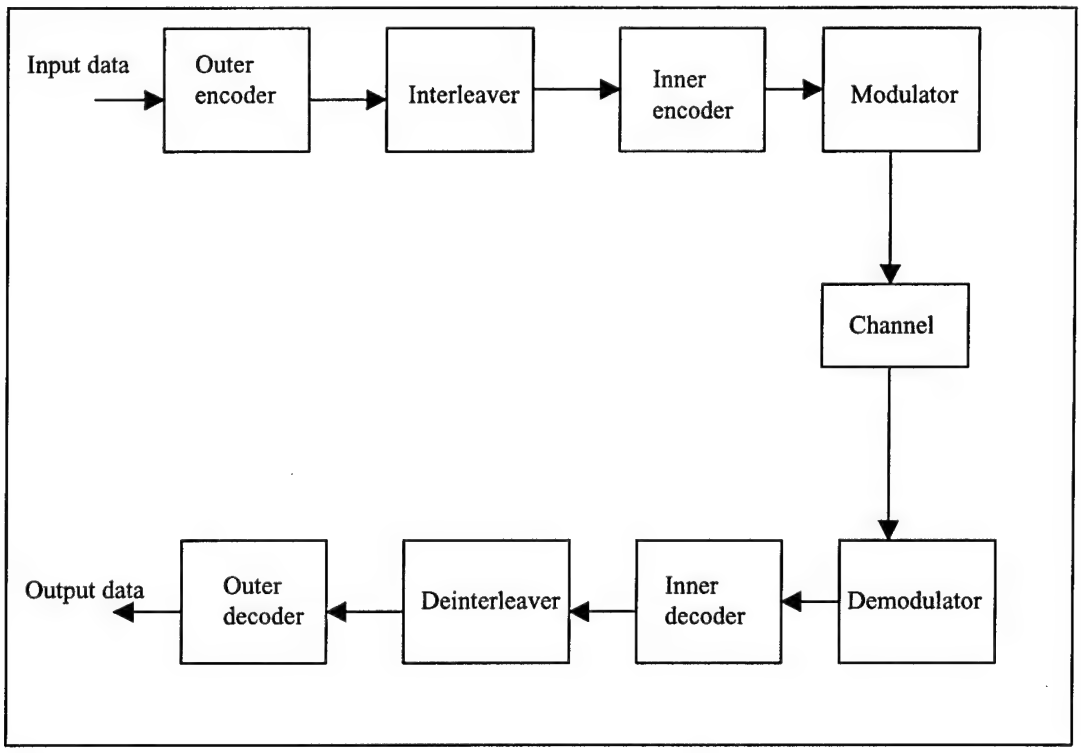

Figure 2-4 Block diagram of a concatenated coding system.

Typical practice has shown that the inner code is designed to be a powerful, lower rate code, while the outer code is a high rate code [22]. Massey [23] stated that convolutional codes should be used in the first stage of decoding because they can easily accept soft decisions and channel state information from the channel. Reed-Solomon codes are then used to clean up the errors left over by the Viterbi decoder. Indeed, Viterbi decoding and the decoding of Reed-Solomon codes complement each other very nicely. The Viterbi decoder has no problems accepting soft decisions from the channel, and it delivers short bursts of errors. Short error bursts do not affect the Reed-Solomon decoder as long as they are within the correction capability of the Reed-Solomon code.

One of the most popular concatenated coding systems uses a Viterbi decoding convolutional code as an inner code and Reed-Solomon as an outer code, with interleaving between the two coding steps [21]. The operation of such systems is such that with E_b/N_0 in the range of 2.0 to 2.5 db the probability of error $p=10^{-5}$ (only about 4 db away from the Shannon limit) can be achieved. This complementary feature was one of the reasons that this concatenated system has been selected by the Consultative Committee for Space Data Systems (CCSDS) of NASA and ESA for deep space missions, where power saving is the main concern [24].

We can summarize the concatenation as a method of constructing long codes from shorter codes. This method was first proposed by Forney [16] in 1966 as a means of constructing long block codes which can be decoded without the complexity of using long codes.

2.4 Turbo Codes

Research in coding theory has seen many proposals aimed at the construction of powerful codes using block and convolutional coding techniques. Shannon theory has proved that larger block length and "random" codes possess good BER. However, the decoding complexity increases exponentially with the block length. On the other hand, the structure imposed on the codes in order to decrease their decoding complexity often results in relatively poor performance. As a result, approaching the channel capacity or even, more modestly, going significantly beyond the channel cutoff rate (practical limit on the highest rate at which a sequence decoder can operate) had been an unreachable dream of coding theorists for many years.

There are two basic approaches to decrease the bit error probability of a system through channel coding. The more traditional approach attempts to increase the minimum Hamming distance of the code. This results in a lowering of the word and bit error probabilities. The goal of the second approach is to reduce the multiplicity (i.e., the number) of code words with low Hamming weights. This is the approach applied to the design of "turbo" codes. Recently proposed 'turbo codes' yield very good performance (near the Shannon limit) in combination with simple iterative decoding strategies. Turbo codes were introduced in 1993 by Berrou, Glavieux, and Thitimajshima [2].

It has been claimed that these codes achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. For a bit error probability of 10^{-5} and code rate of 1/2, the authors report a required E_b/N_0 of 0.7 dB and block lengths of 65,536. The codes are constructed by applying two or more

constituent codes to different interleaved versions of the same information sequence. Decoding calls on iterative processing in which each component decoder takes advantage of the work of the other at the previous step. Since the introduction of turbo codes, numerous research efforts have been dedicated to optimize the components of turbo codes and their applications to wireless mobile communication systems.

2.4.1 Turbo Code Encoder

A block diagram of the encoder of turbo code is shown in Figure 2-5. As seen in the figure, a turbo encoder consists of two parallel concatenated convolutional encoders called constituent codes (Recursive Systematic Convolutional encoders (RSC)) separated by an interleaver, with an optional puncturing mechanism.

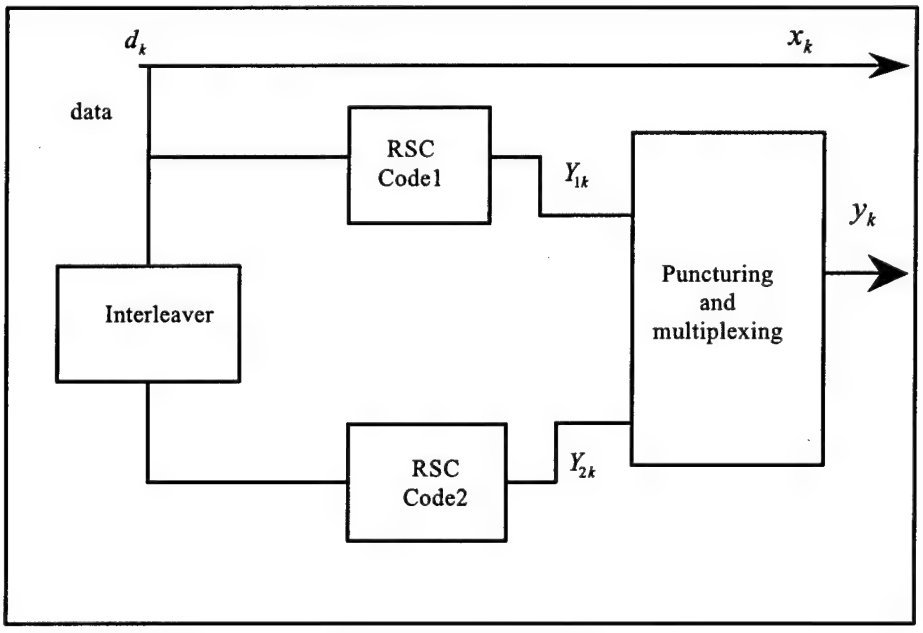

Figure 2-5 Turbo code encoder.

2.4.1.1 The Constituent Encoder

The first block of the turbo code encoder is the constituent encoder. A distinctive feature of turbo codes is the fact they use systematic recursive convolutional codes [25, 26].

The most typical form of a convolutional encoder is the nonrecursive nonsystematic convolutional (NSC) encoder [11]. For a rate 1/2 convolutional encoder, the constraint length is K and memory is K-1. The input to the encoder at time k is a bit d_k, and the corresponding code word is the bit pair (u_k, v_k), where:

$$u_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i} \qquad \text{modulo-2} \qquad g_{1i} = 0,1 \tag{2-17}$$

$$v_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i} \qquad \text{modulo-2} \qquad g_{2i} = 0,1 \tag{2-18}$$

$G_1 = \{g_{1i}\}$ and $G_2 = \{g_{2i}\}$ are the code generators, generally expressed in octal form, and d_k is represented as a binary digit. This encoder has a finite impulse response (FIR). An example of NSC code is shown in Figure 2-6.

In this example, the constraint length is K= 5, and the two generators are (in octal form) G_1 = 37 and G_1 = 21. It is well-known that the BER of a classical NSC code is lower than that of a classical Systematic code (SC) with the same memory at large signal-to-noise ratio (SNR). At low SNR, the performance of NSC and SC are generally reversed.

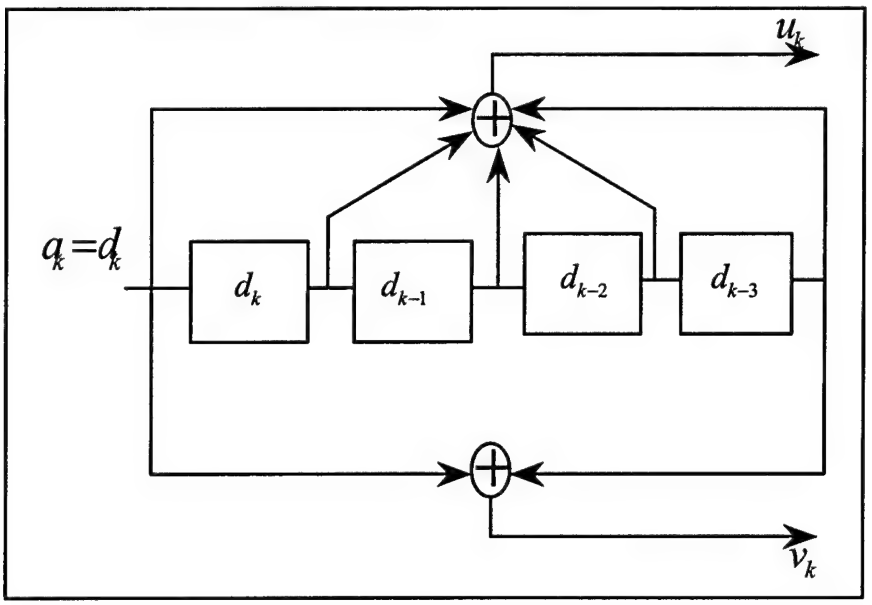

Figure 2-6 Classical nonsystematic convolutional code.

The new class of Recursive Systematic Convolutional (RSC) codes, the infinite impulse response (IIR) convolutional codes, has been proposed in [2, 27]. Performance of IIR codes can be shown to be better than the best NSC code at any SNR for high code rates larger than 2/3. A binary rate 1/2 RSC code is obtained from a NSC code by using a feedback loop, and setting one of the two outputs (u_k or v_k) equal to d_k. Figure 2-7 illustrates an example of RSC code, with K = 5, where a_k is recursively calculated as follows:

$$a_k = d_k + \sum_{i=1}^{K-1} g_i^* a_{i-1} \quad \text{modulo-2} \tag{2-19}$$

where g_i^* is respectively equal to g_{1i} if $u_k = d_k$, and to g_{2i} if $v_k = d_k$. It is stated in [2] that a_k exhibits the same statistical properties as d_k. The trellis structure is identical for the RSC code of Figure 2-7 and the NSC code of Figure 2-6, and these two codes have the same free distance (free distance is the minimum distance in the set of all arbitrarily long paths that diverge and re-emerge in the trellis diagram). However, the two output sequences $\{u_k\}$ and $\{v_k\}$ do not correspond to the same input sequence $\{d_k\}$ for RSC and NSC codes. For the same code generators, RSC codes do not modify the output weight distribution of the output codewords compared to NSC codes. They only change the mapping between input data sequences and output codeword sequences.

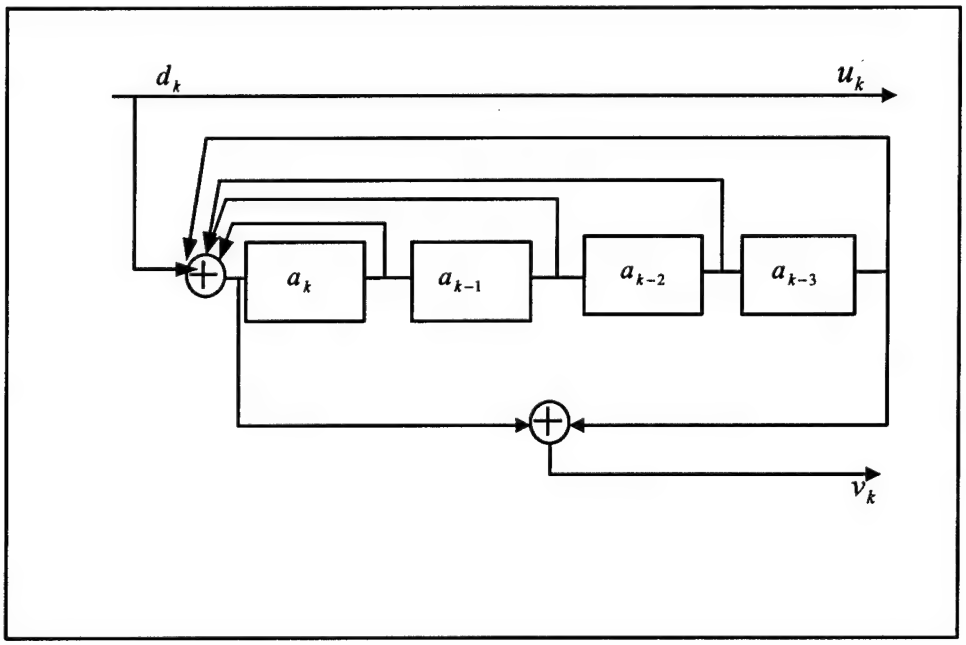

Figure 2-7 Recursive systematic convolutional code.

In Figure 2-8, a parallel concatenation of two RSC encoders of the type shown in Figure 2-7, provides an example of a rate R = 1/2 turbo code encoder. Two encoders with $K = 4$, and the same polynomial generators (37, 21) are used in [2]. Both encoders receive the same information sequence $\{d_k\}$ but are arranged in a different sequence due to the presence of the interleaver. Puncturing (some outputs bits X_k or Y_k are deleted according to a chosen pattern) is often used to increase the rate to 1/2. Without the switch (puncture), the code would be rate 1/3. Additional concatenation can be continued with multiple component codes. In general, the component encoders need not be identical with regard to constraint length and rate.

Choosing the best constituent code is one way of improving the performance of turbo codes. Different researchers take different optimum criteria to optimize the constituent codes. In [28], the search for the best constitute encoder is based only on maximizing the input weight 2 free distance of the code (minimal output weight codewords most of time due to the input weight 2, the importance of input weight 2 is explained in weight distribution of turbo codewords section). In [29] the search is based on maximizing codewords of input weight 2 and 3 (d_2 and d_3). In [30], the authors examined d_2, as well as minimized the number of nearest neighbors (N_{eff}), where N_{eff} is the number of paths having the same effective distance. In [31], the search is based on optimization of the pairs (d_i, N_i), where d_i represent the input sequence of weight i and N_i are their multiplicities (i.e. the number of code sequences of weight d_i generated by input sequences of weight i). The authors in [31] performed sequential optimization. First d_i is maximized and then N_i minimized from $i = 2$ to $i = 6$. The authors take into account the multiplicity of low weight sequences which lead to significantly better codes.

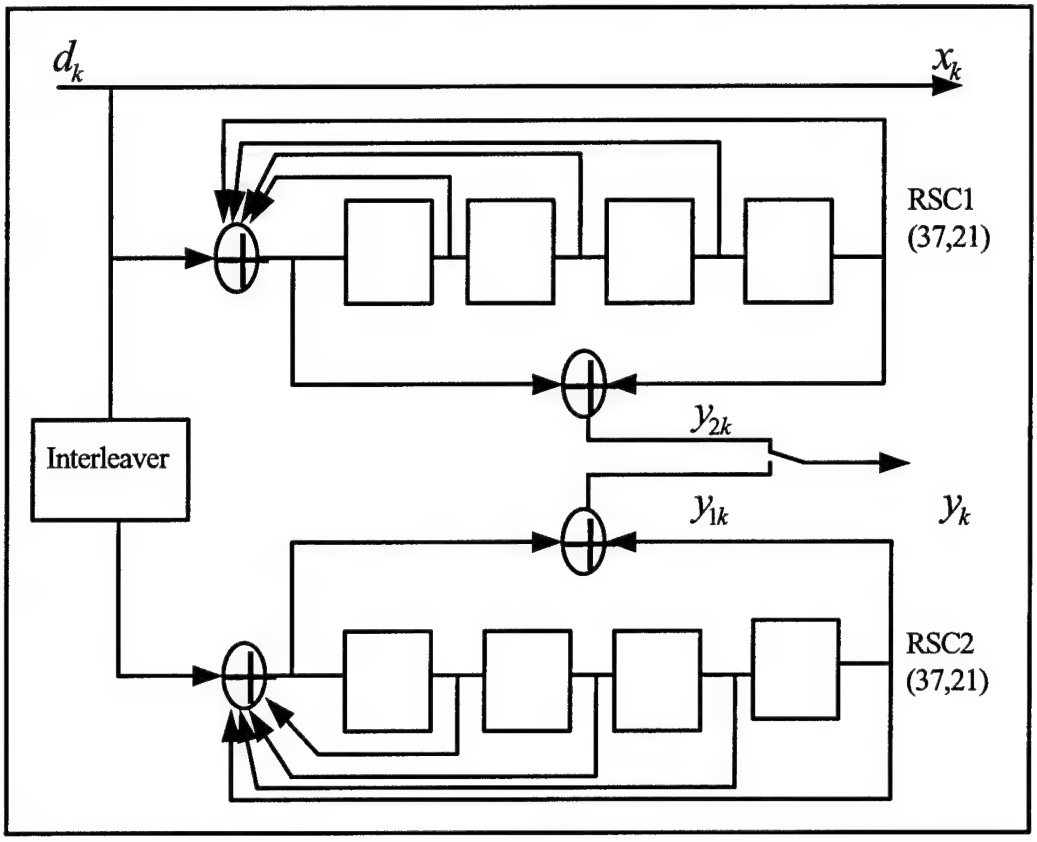

Figure 2-8 Recursive systematic convolutional codes with parallel concatenation.

The encoder in Figure 2-8 is used to generate a $(2(N+M), N)$ block code, where N is the information size. Following the information bits, additional $M = K-1$ tail bits (K is the constraint length of the convolutional code) are appended in order to drive the encoder to the all-zero state at the end of the block. Due to the encoders' recursiveness, the required M tail bits cannot be predetermined automatically. This action of returning the encoders to the all-zero state is called trellis termination. Some input data sequence of length N are self-terminating because the encoders are already in the all-zero state after

encoding N information bits and M tail bits are appended. All M tail bits are zero for a self-terminating input sequence. Non-self-terminating input sequences require one or more nonzero tail bits for proper trellis termination.

Since the component encoders are recursive, it is not sufficient to set the last M information bits to zero in order to drive the encoder to the all-zero state (i.e., to terminate the trellis). The termination tail sequence depends on the state of each component encoder after N bits. Due to the presence of the interleaver, it is impossible to simultaneously terminate the trellises of both component encoders using the same M bit tail [32]. The solution to the problem of terminating the trellis of component encoders has not been stated in the original paper [2]. In [33, 34], it is shown that by properly designing the interleaver, it is possible to force both constituent encoders back to the all-zeros state with a single M bit tail.

Some researchers [35, 36] suggest terminating only one encoder. These studies take differing approaches by either terminating the first encoder or terminating the second encoder. The authors in [37] provide a survey for the termination schemes used for short frames. The authors conclude that at low bit error rates, the termination errors dominate, where perfect termination (terminate both encoders and send both tailing bits) has fewer errors. In [38], the authors proposed a new method for termination. This method is not included in the survey [37]. This method of termination uses both encoders and sends the systematic tailing bits of the first encoder and the check bits of both encoders. The approach uses the configuration in Figure 2-9 to terminate both encoders. In our simulation model we used this scheme for terminating both encoders.

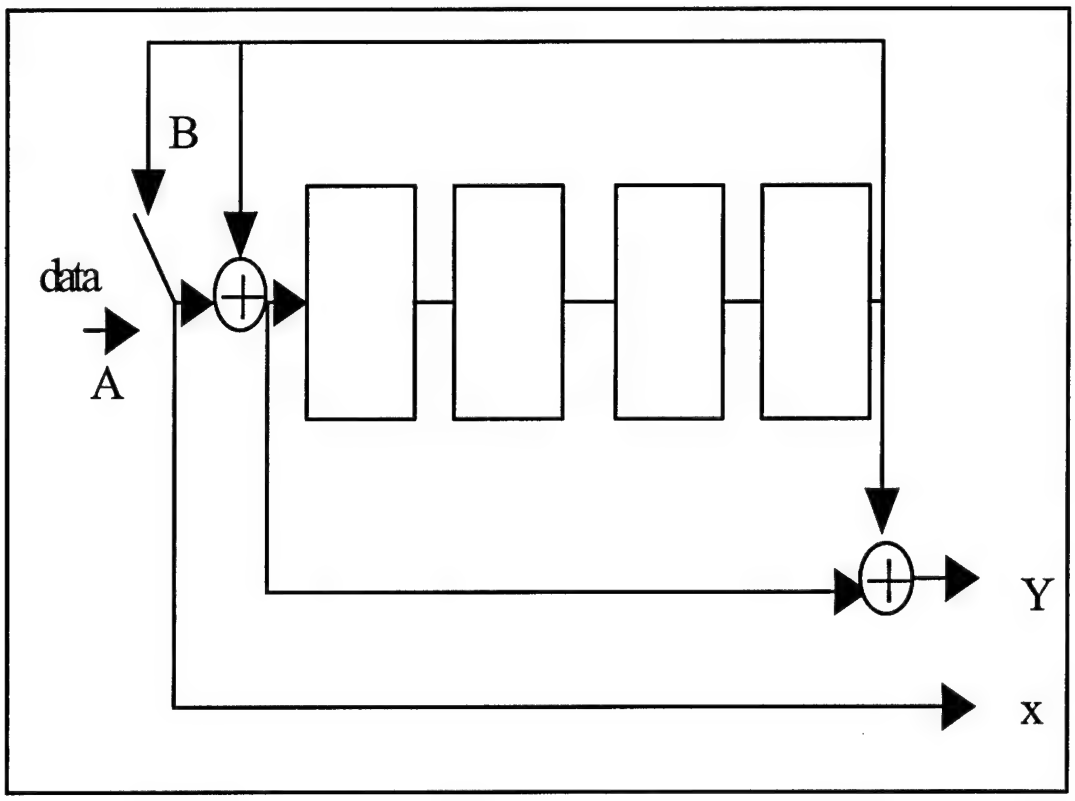

Figure 2-9 Trellis termination scheme.

The configuration in Figure 2-9 is sufficient to terminate the trellis at the end of the block, where the switch is in position "A" for the first N clock cycles (N is the number of information bits per frame) and is in position "B" for the additional required tailing bits, which will flush the encoder with zeros. The same termination is used for both encoders.

2.4.1.2 Interleaver

The turbo encoder in Figure 2-8 produces codewords from each of two component encoders. The weight distribution for the codewords out of this parallel concatenation depends on how the codewords from one of the component encoders are combined with codewords from the other encoder(s). Good code design attempts to avoid pairing low-weight codewords from one encoder with low-weight codewords from the other encoder, to increase the minimum weight of the code words. The trick in turbo coding is to match low-weight codewords of one encoder with high-weight codewords from the other encoder(s). This results in total weights significantly higher than the low weights that are possible from each of the simple component codes individually. Many such pairings can be avoided by proper design of the interleaver [38-42]. One important property of turbo code is that its minimum distance is not fixed by the constituent codes but by the interleaving function. Finding the optimum interleaver for turbo code remains a real challenge [27].

The choice of the interleaving scheme also has an effect on the performance of the overall code. An interleaver that permutes the data in a random fashion provides better performance than the familiar block interleaver [41]. For short frames, the performance of block interleaving is quite close to the best nonuniform interleavers; the difference between interleavers becomes clear only at bit error rates lower than 10^{-3} [43]. Another way to choose the interleaver is to combine the problem of trellis termination with interleaving. By using a "simile" type interleaver proposed in [33], the trellis of both constituent encoders can be terminated with one set of tail bits. In [38], the authors

proposed a new interleaver, known as S-random interleaver. For a given S, this interleaver satisfies the constraint that any two symbols separated by fewer than S symbols in the input sequence are separated by at least S symbols in the interleaved sequence.

2.4.2 Turbo Decoder

In digital transmission systems, the received signal is a sequence of waveforms whose correlation extends well beyond the signaling period (T_s). There can be many reasons for this correlation, such as coding, intersymbol interference, or correlated fading [44]. It is well-known that the optimum receiver in such situations cannot perform its decisions on a symbol-by-symbol basis. Deciding on a particular information symbol, u_k, involves processing a portion of the received signal T_d seconds long, with $T_d > T_s$. The decision rule can be either optimum with respect to the individual symbol, u_k, $k = 1,2, \ldots,n$, where n is the number of sequence symbols in sequence U, or with respect to the sequence of symbols $U = (u_1, u_2, \ldots,u_n)$. The ultimate goal of the decoding operation is to minimize the errors in the finally decoded sequence $\hat{U}$. Two optimum decoding criteria are thus proposed; the minimum symbol error rate and the minimum word (sequence) error rate. When U is a binary sequence, the minimum symbol error rate criterion achieves the minimum bit error rate (BER). A minimum symbol error rate decoder maximizes the a posteriori probability $P(\hat{u}_k|Y)$, where Y is the received sequence, i.e.,

$$P(\hat{u}_k|Y) = \max_{all\ u_k} P(u_k|Y) \tag{2-20}$$

where $\hat{u}_k$ is the maximum a posteriori probability (MAP) estimate of symbol u_k, $k = 1, \ldots, n$. Similarly, a minimum word error rate decoder maximizes the a posteriori probability $P(\hat{U}|Y)$, i.e.,

$$P(\hat{U}|Y) = \max_{all\ U} P(U|Y) \tag{2-21}$$

where $\hat{U}$ is the MAP estimate of the information sequence U. It has been shown that the performances of these decoders are almost identical in any error criterion [45]. The Viterbi algorithm [46] is an optimal decoding method for minimizing the probability of word error rate.

Optimum symbol decision algorithms have been known since the early 1970s [47, 48]. These algorithms are much less popular than the Viterbi algorithm and almost never applied in practical systems. There is a very good reason for this neglect. They yield performance in terms of symbol error probability only slightly superior to the Viterbi algorithm, yet they present a much higher complexity [44].

A maximum likelihood (ML) decoding of turbo code is impractical because of the computational complexity. By taking advantage of the structure of the turbo code, the decoding can be broken into simpler decoding steps. This allows for the total decoding complexity to be lower than that of the ML algorithm. A known suboptimal decoder for the turbo code is the iterative decoder [2]. From the results reported in [2, 38], it is deduced that the iterative decoder achieves a performance that is close to the Shannon

limit (and consequently, is close to the performance of the ML decoding after few iterations).

The turbo decoder is constructed from simple constituent decoders which share bit reliability measures. The constituent decoder is the optimal decoder for the component codes used by the turbo encoder. The decoder depicted in Figure 2-10 is made up of two elementary decoders (DEC1 and DEC2) in a concatenated scheme [2].

The first elementary decoder DEC1 is associated with the first encoder and yields a soft decision. The error bursts at DEC1 output are scattered by the interleaver.

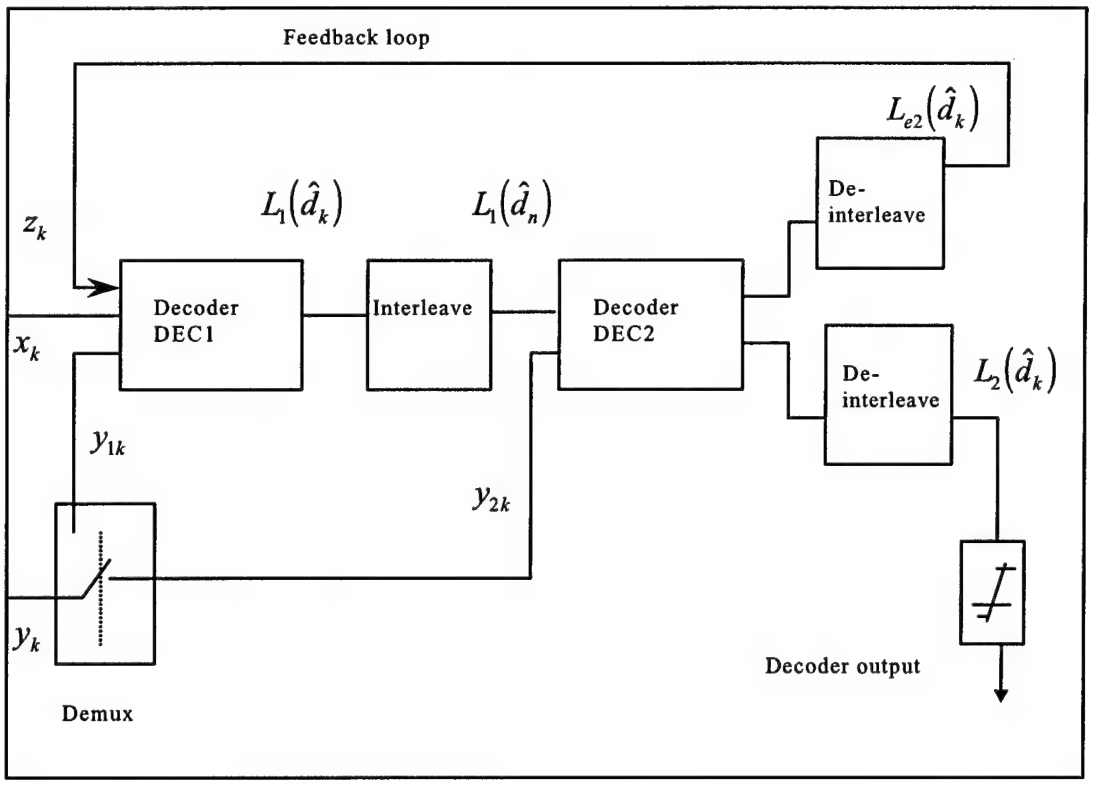

Figure 2-10 Turbo decoder.

For a discrete memoryless Gaussian channel and a binary modulation, the decoder input is made up of two random variables x_k and y_k, at time k:

$$x_k = (2d_k - 1) + i_k \tag{2-22}$$

$$y_k = (2Y_k - 1) + q_k \tag{2-23}$$

where i_k and q_k are two independent noises with the same variance σ^2. The redundant information y_k is demultiplexed and sent to the decoder DEC1 when $Y_k = Y_{k1}$ and toward decoder DEC1 when $Y_k = Y_{k2}$. The first decoder DEC1 must deliver to the second decoder DEC2 a soft decision. The logarithm of likelihood ratio (LLR), $L'(d_k)$ associated with each decoded bit d_k by the first decoder DEC1 is used as a relevant piece of information for the second decoder DEC2, where:

$$L'(d_k) = \log\left[\frac{P\{d_k = 1|Y\}}{P\{d_k = 0|Y\}}\right] \tag{2-24}$$

where $P\{d_k=1|Y\}$, $i = 0, 1$ is the a posteriori probability (APP) of the bit d_k given knowledge of the received data Y.

$$L'(d_k) = \log\left[\frac{P\{Y|d_k = 1\}\ P\{d_k = 1\}}{P\{Y|d_k = 0\}\ P\{d_k = 0\}}\right]$$

$$= \log\left[\frac{P\{Y|d_k = 1\}}{P\{Y|d_k = 0\}}\right] + \log\left[\frac{P\{d_k = 1\}}{P\{d_k = 0\}}\right]$$

$$L'(d_k) = L_c(d_k) + L(d_k) \tag{2-25}$$

where $L_c(d_k)$ is the LLR of the channel measurements of d_k and $L(d_k)$ is the a priori LLR of the d_k. Equations 2.24 and 2.25 were developed with only the data detector in mind [49]. The introduction of a decoder typically yields decision-making benefits. For a systematic code, it can be shown that the LLR (soft output) $L(\hat{d}_k)$ out of the decoder is equal to:

$$L(\hat{d}_k) = L'(\hat{d}_k) + L_e(\hat{d}_k) \tag{2-26}$$

where $L'(\hat{d}_k)$ is the LLR of the data bit out of the detector (input to the decoder), and $L_e(\hat{d}_k)$, called the extrinsic LLR, represents extra knowledge that is gleaned from the decoding process. The output sequence of a systematic decoder is made up of values representing data and parity. Equation 2-26 partitions the decoder LLR into the data portion represented by the detector measurements, and the extrinsic portion represented by the decoder contribution due to parity. From Equations 2-25 and 2-26:

$$L(\hat{d}_k) = L_c(d_k) + L(d_k) + L_e(\hat{d}_k) \tag{2-27}$$

The soft decision $L(\hat{d}_k)$ is a real number that provides a hard decision as well as the reliability of that decision. For iterative decoding, the extrinsic likelihood is fed back to the decoder input to serve as a refinement of the a priori value for the next iteration.

The decoding algorithm for turbo codes uses an iterative procedure whose heart is an algorithm which computes the sequence of the a posteriori probability (APP) distributions of the information symbols. Unfortunately, the Viterbi algorithm is not able to yield the APP for each decoded bit. A relevant algorithm for this purpose has been

proposed by in [47], and is also known as the MAP algorithm. This algorithm minimizes the bit error probability in decoding linear block and convolutional codes and yields the APP for each decoded bit. For RSC codes, the MAP algorithm has been modified in [2] in order to take into account their recursive character. However, the MAP decoder suffers from a complexity that is significantly higher than that of the Viterbi algorithm [2]. Currently, effort is underway to find reduced complexity decoders that can produce soft outputs. Two approaches have been taken. The first approach tries to modify the Viterbi algorithm to yield soft output. This is known as the Soft-Output Viterbi Algorithm (SOVA) [50-52]. Because the MAP algorithm suffers from a complexity that is significantly higher than that of the Viterbi algorithm, the second approach consists of revisiting the original symbol MAP decoding algorithm with the aim of simplifying it to a form suitable for implementation [53-57].

2.4.3 Performance Analysis of Turbo Codes

Many research efforts on turbo codes have been published seeking to find the exact explanation for its extraordinary performance and to provide methods for further improvement. Since the output weight distribution decides the performance of turbo codes, these efforts are concentrated on finding the connection between the output weight and the system components.

2.4.3.1 The Weight Distribution of Turbo Codes

In order to estimate the performance of a code, it is necessary to have information about its minimum distance, d_{min}, weight distribution, or actual code geometry, depending on the accuracy required for the bounds or approximations. However, the minimum

distance is not the most important quantity of the code, except for its asymptotic performance at very high E_b/N_0. At moderate signal-to-noise ratios (SNRs), the weight distribution at the first several possible weights is necessary to compute the code performance. Estimating the complete weight distribution for a large N is still an open problem for these codes. The proper choice of the interleaver can reduce the number of low-weight codewords.

The weight distribution for the codewords produced by the turbo encoder depends on how the codewords from one of the simple component encoders are teamed with codewords from the other encoder. Due to recursiveness of the encoders, it is important to distinguish between self-terminating and non-self-terminating input sequences. The non-self-terminating sequences have little effect on decoder performance. This is due to the accumulated high encoded weight that caused the code to artificially terminate at the end of the block. From probabilistic arguments based on selecting the permutations randomly, it is concluded that the self-terminating weight-2 data sequences are the most important consideration in the design of the constituent codes [41]. There are also many weight-n, n=3, 4, 5, . . ., data sequences that produce self-terminating output and hence low encoded weight. However, these sequences are much more likely to be broken up by the random interleavers than the weight-2 sequences [38]. Higher-weight, self-terminating sequences have successively decreasing importance. Better weight distributions were obtained by using random interleaver [39, 41]. One can argue that turbo code performance is determined largely from minimum weight codewords that result from the weight-2 input sequence.

2.4.3.2 The Analytical Bound of Turbo Code

The discovery of turbo codes and the near capacity-performance reported in [2] has stimulated numerous research efforts to fully understand this new coding scheme. Initially, the original results were independently reproduced by several researchers [58-60]. Subsequently, recent research [28, 41, 42, 58, 61] on turbo codes has focused on understanding the reasons for their outstanding performance.

Despite various attempts, a satisfactory explanation yielding a good comprehension of the behavior of turbo codes has not yet appeared in literature. Only cut-and-try approaches based on simulation have been addressed to the problem of designing the interleaver, as well as to that of performance evaluation.

Since the first appearance of turbo codes, many of the structural properties of turbo codes have now been on firm theoretical footing [59, 61, 62]. In [62], the authors derive an analytical upper bound to the average performance of such a coding scheme. The average upper bound was constructed by averaging over all possible interleavers. This is independent of the interleaver used and shows the influence of the interleaver length on the code performance.

It is useful to consider turbo codes as block codes. The input sequences are restricted to length N, where N corresponds to the size of the interleaver in the turbo encoder. Consider a traditional union upper bound for maximum likelihood decoding of a (N, K) block codes. Without loss of generality, it is assumed that the all-zero codeword was sent, and the upper bound on the probability of word error is [63]:

$$P_e \leq \sum_{d=d_{free}}^{N} A(d)\ P_2(d) \tag{2-28}$$

where $A(d)$ denotes the number of codewords that have Hamming weight d. $P_2(d)$ is the pairwise error probability between code word with weight "0" and the codeword with weight "d", and d_{free} is the minimum weight codeword different from the all-zero codeword.

For a turbo code with a fixed interleaver, the construction of $A(d)$ requires an exhaustive search. Due to complexity issues involved in this search, [62] proposes an average upper bound constructed by averaging over all possible interleavers. The result of this averaging can be thought of as the traditional union upper bound, but with an average weight distribution. As in [62], the average weight distribution can be written as

$$\overline{A(d)} = \sum_{i=1}^{K} \binom{K}{i} P(d|i) \tag{2-29}$$

where $\binom{K}{i}$ is the number of input words with Hamming weight i and $P(d/i)$ is the probability that an input word with Hamming weight i produces a codeword with Hamming weight d. Substituting into Equation 2-29, the average upper bound for word and bit error can be expressed as:

$$P_e \leq \sum_{d=d_{free}}^{N} \overline{A(d)}\ P_2(d)$$

$$= \sum_{d=d_{free}}^{N} \sum_{i=1}^{K} P(d \mid i) P_2(d)$$

$$= \sum_{i=1}^{K} \binom{K}{i} E_{d/i}\left[P_2(d)\right] \tag{2-30}$$

Then, the probability of bit error can be written as follows:

$$\overline{P_{bit}} = \sum_{i=1}^{K} \frac{i}{K} \binom{K}{i} E_{d/i}\left[P_2(d)\right] \tag{2-31}$$

In Equations 2-30, $E_{d/i}[.]$ is an expectation with respect to the distribution $P(d/i)$. This average upper bound is attractive because relatively simple schemes exist for computing $P(d/i)$ from the state transition matrix of the RSC [62]. The performance of turbo codes can be studied on various statistical channels by formulating the two-codeword probability $P_2(d)$ for the channel of interest and using Equation 2-30 or 2-31.

Examining the analytical bound of turbo codes in the Additive White Gaussian Noise (AWGN) channel yields the following conclusions. The region where the turbo codes have offered astounding performance is below the computational cutoff rate threshold. So at first glance, the bounds appear to be of dubious utility. For E_b/N_0 above the computational cutoff rate threshold, the bound is not only meaningful but it essentially tells the whole story (i.e., the bit-error rate predicted by the bound is accurately achieved both by a maximum-likelihood decoder and by a turbo decoder) [62].

In [64], the performance of turbo codes is addressed by examining the code's distance spectrum. The "error floor" that occurs at moderate signal-to-noise ratios is shown to be a consequence of the relatively low free distance of the code. It is also shown that the "error floor" can be lowered by increasing the size of the interleaver

without changing the free distance of the code. Alternatively, the free distance of the code may be increased by using primitive feedback polynomials. The excellent performance of turbo codes at low signal-to-noise ratios is explained in terms of the distance spectrum. Thus it can be concluded that the outstanding performance of turbo codes at low signal-to-noise ratios is a result of the dominance of the free-distance asymptote. This is a consequence of the sparse-distance spectrum of turbo codes, as opposed to spectrally dense convolutioal codes. Finally, the sparse-distance spectrum of turbo codes is due to the structure of the codewords in a parallel concatenation and the use of pseudorandom interleaving. The theory of spectral thinning is introduced and formulated in [64] and used to explain the performance of turbo codes at low SNR's.

From the above discussion, it is seen that the two main tools for the performance evaluation of turbo codes are computer simulation and union bound. The union bound diverges for very low signal-to-noise ratios and is useless below the channel cutoff rate. Computer simulation is, therefore, the only method for the performance evaluation of turbo codes below channel cutoff rate. In [62], the authors stated the possibility of applying the Gallager bound (tight bound) to improve the union bound. In [65], the authors derive a new upper bound on the word and the bit error probabilities for turbo codes with maximum likelihood decoding. The new bound is tight for the large range of signal-to-noise ratios extending below the channel cutoff rate, which is the region where the union bound diverges. The derivation of this bound is based on the application of the Gallager bound [63].

2.4.4 Turbo Codes in Wireless Communication Systems

In a multiple-access scheme like CDMA, the capacity (maximum number of users per cell) can be expressed as $C = \frac{\eta}{E_b/N_0} + 1$, where η is the processing gain and E_b/N_0 is the required signal-to-noise ratio to achieve a desired bit error rate (BER) performance [38]. For a given BER, a smaller required E_b/N_0 implies a larger capacity. Unfortunately, to reduce E_b/N_0 it is necessary to use very complex codes (e.g., large constraint length convolutional codes). Turbo codes promise to achieve superior performance with limited complexity in mobile communication systems. For example, if a (7, 1/2) convolutional code is used at BER = 10^{-3}, the capacity is $C = 0.5\eta$. However, if two (5, 1/3) punctured convolutional codes or three (4, 1/3) punctured codes are used in a turbo encoder structure, the capacity can be increased to $C = 0.8\eta$ (with 192-bits and 256-bits interleavers which correspond to 9.6 Kbps and 13 Kbps with roughly 20 ms frames) [38].

Turbo codes are based on the encoding of finite data frames. The error performance of turbo codes is mainly determined by the number of bits per data frame. In order to successfully use turbo codes in mobile radio, the following two main requirements must be met [66]. First, restrictions apply to the maximum delay in speech transmission introduced by the interleavers. In speech transmission, which is typically based on finite frames, the speech frames contain less than 200 bits to be processed by the channel encoder. For instance, the speech frames contain 189 bits in the case of the pan-European Global System for Mobile Communications (GSM) and Digital Cellular

System (DCS) 1800 [20], and 192 bits in the uplink of the recently introduced joint detection code division multiple access (JD-CDMA) mobile radio system mobile radio system [67]. JD-CDMA represents an evolution of time division multiple access (TDMA) based mobile radio systems such as GSM, DCS 1800 and the third generation proposal advanced TDMA (ATDMA) [68]. Mobile radio systems suffer from such delays caused by the interleavers. In order not to introduce a further delay penalty, turbo codes must take into account the size of the speech frames. For large frame sizes (>200 bits), the random interleavers are preferred [69]. Unlike large frame systems, block interleavers seem to be suitable for turbo code encoders in short frame transmission systems. However, for speech transmission requiring a BER of ~ 10^{-3}, apparently any interleaver appears to be applicable [70]. In [71], it has been shown that, by using a structured interleaver, instead of a random interleaver, an increased minimum Hamming distance can be obtained. In [71], the author showed that the short interleaver introduces a short decoding delay where, in the case of a turbo code, a short interleaver may have an additional advantage. Since a soft-in/soft-out decoding algorithm should be implemented for the constituent codes, short interleavers may reduce the decoding complexity in terms of the required memory.

The application of turbo codes to a TDMA/CDMA mobile radio system was investigated in [72]. Gains of 0.4 to 1.2 dB over non-systematic convolutional codes can be achieved by using turbo codes for the considered mobile radio system using joint detection with coherent receiver antenna diversity. High spectral efficiency modulation schemes using turbo codes for AWGN and Rayleigh channels were presented in [73]. An AWGN channel was assumed with an encoder using a non-uniform interleaver (64 × 64)

and 16 QAM modulation. Three iterations were used in the decoder. A BER of 10^{-6} was achieved at an $E_b/N_0 = 4.35$ dB for 2 bit/s/Hz spectral efficiency and at an $E_b/N_0 = 6.2$ dB for 3 bit/s/Hz spectral efficiency. For a Rayleigh channel with four decoder iterations, a BER of 10^{-5} was achieved at an $E_b/N_0 = 6.5$ dB for 2 bit/s/Hz spectral efficiency and an $E_b/N_0 = 9.6$ dB for 3 bit/s/Hz spectral efficiency.

In [36], it was concluded that for BER > 10^{-3} and a large frame error rate (FER) of 10^{-2}, the effect of the chosen interleaver on the performance of the corresponding turbo code is almost negligible in the case of short transmission (192 bits). However, for a slightly larger block size (399 bits), the interleaver type can be optimized to double the performance of the turbo code [69].

In [74], the BER and the FER for short frame transmission (192 bits) over an AWGN channel was investigated. The best pseudo-random interleaver from about 800 tested was selected, and a MAP estimator as in [2] was used with proper termination of the second code. A BER of 1.2×10^{-3} and a FER of 2.5×10^{-2} was achieved at an $E_b/N_0 =$ 2.0 dB after 10 iterations. These results are reported to be 1.2 to 1.7 dB better than the performance achieved with a non-systematic convolutional code with the same memory order.

Besides the short frames that characterize the speech transmission in mobile radio systems, the propagation channel characteristics have an important role in the performance and design of the turbo codes. The mobile radio propagation channel, as mentioned before, is a hostile environment. Multipath, interference, and noise all contribute to degrade signal quality at the receiver. To date, only limited attention has

been given to the performance of turbo codes on fading channels [73-75]. In [73], the authors presented a simulation of a new coding scheme, using the association of a turbo code with a quadrature amplitude modulation for both Gaussian and Rayleigh channels, using 64 × 64 nonuniform interleaver. The authors reported that the performance of the turbo code with QAM is always better than the 64-state Trellis Code Modulation (TCM) at BER values less than to 10^{-3}. In [74], the author discusses a typical turbo code with rate 1/2 using two identical component codes with octal generators 15, 17 and constraint length 4 (memory 3), and structured 12 × 16 block interlever (192 bit-frames). Furthermore, a novel low complexity turbo-code decoder which uses a simplified and thus suboptimum version of the MAP, was introduced in [74]. In the simulation, the second RSC code was terminated. This allows for better error performance at E_b/N_0 than the case of termination for the first RSC code [32, 36]. The author compared the proposed new decoder with other decoders [2, 35], in AWGN and fully interleaved Rayleigh fading channels. It was shown that the application of the novel low complexity turbo-code decoder is attractive. The author again used the same typical turbo code proposed in [74] to apply it to JD-CDMA Mobile Radio system using coherent receiver antenna diversity [67]. In [75], a typical turbo code was considered with rate 1/3 using the 16-state RSC (memory 4) with generator in octal form $(21/37)_8$. In the simulation, the authors use different block sizes of 420, 5000, and 50 000 bits. In all simulations, the turbo decoder uses the MAP algorithm with modifications found in [32]. For $N = 420$, a helical interleaver was used which has been shown to be effective on the AWGN channel [33]. The performance and design of turbo codes using coherent BPSK signaling on the Rayleigh fading channel has also been considered. In low signal-to-noise regions,

performance analyses use simulations of typical turbo coding systems. For higher signal-to-noise regions, an average upper bound is used in which the average is over all possible interleaving schemes. Fully interleaved and exponentially correlated Rayleigh channels are explored.

Application of turbo code combined with the bandwidth efficient method, Trellis-Coded Modulation (TCM), has been presented in [76]. This combination has been implemented as a simple modification of pragmatic TCM. This new system achieved coding gains of 1-2 db in simulation results relative to pragmatic TCM. These gains were achieved with comparable complexity but with greater delay than with pragmatic TCM.

Trellis coded modulation is a standard method to prevent bandwidth expansion and achieve relatively high coding gains. The authors in [77] combine the power of turbo coding techniques with the bandwidth of trellis coded modulation to obtain the turbo code modulation scheme. In [77], the authors provide a method for applying the standard union bound for turbo code modulation systems. In particular, they explicitly derive the bound for a 2-bits/s/Hz 16 QAM turbo code modulation system. The suggested method can easily be adopted for other turbo code modulation systems. The derived bound provides a tool for comparing coded modulation schemes having different component codes, interleaver lengths, mapping, using maximum likelihood decoding. It is also useful in studying the effectiveness of various suboptimal decoding algorithms. The authors in [77] have extended their methodology to the fading channel in [78].

Turbo codes can also offer unequal error protection (UEP). The Rate Compatible Punctured Convolutional (RCPC) [79] can be extended to turbo codes to achieve unequal

error protection. Rate Compatible Turbo Codes have been considered to achieve unequal error protection in [80]. This allows turbo codes to be used for channels where some information bits are more sensitive to errors than the others at the same time, where powerful codes are required like speech or image in the mobile environments. Rate Compatible Punctured turbo code (RCPTC) has been introduced in [80-84] to be used in FEC/ARQ systems for data transmission.

2.5 Summary

In this chapter the parameters and types of fading in wireless communication systems were reviewed. The channel coding techniques deployed in wireless communication systems; block codes, convolutional, and concatenated codes have been presented.

The performance of turbo code is sensitive to its code structure, which comprises the code rate, constraint length, tap connection, block size, interleaving pattern, and number of decoding iterations. This chapter also addressed the problem of choosing the different components of turbo codes and how this choice can affect the performance of turbo codes. The problem of the application of turbo codes to cellular mobile communication systems was also addressed

From this discussion, a conclusion can be made that the optimizing of different components of turbo codes are very related to the application. Choosing the constituent encoder depends on its distance spectrum. If the quality of service needed requires low bit error rate (like data transmission), then the focus is on choosing a constituent encoder with sparse spectrum at the first few low weight codewords. Here, the termination of

both encoders is important. In cases where lower signal-to-noise ratios or higher bit error rates are permissible (like speech transmission), the distance spectrum of constituent encoders plays a very important role in choosing this component. In this region, the number of multiplicity of low and moderate weight codewords is required to be as minimal as possible. Also choosing the length of the interleaver is restricted by the application sensitivity to the time delay. This restricts the performance in such cases (like speech transmission). Performance of the interleaving map depends on the length of the interleaver. For short frames, the structured interleaver do better than the nonstructured one, but for longer interleaver the random interleavers are the best.

Also the sensitivity to the delay will restrict the number of iterations in the decoder. In addition, the complexity and the size of the memory available will restrict the component decoder type used.

In the next chapter, the principle of the iterative decoding process of turbo code decoder will be presented. The Viterbi algorithm and its modified version Soft Output Viterbi Algorithm (SOVA) will be presented.

3 Turbo Decoder

3.1 Introduction

In the previous chapter, the background of turbo code with two parallel concatenated covolutional codes and its related components were presented. In this chapter, the principle of the iterative decoding process of the turbo code decoder is presented. Turbo code decoder consists of two soft-input/soft-output constituent convolutional decoders that works together in an iterative fashion. Both constituent decoders accept soft input and deliver soft output. Section 3.2 presents the principle of the iterative decoding process of two-dimensional systematic convolutional codes using any soft-input/soft-output constitute decoder. Section 3.3 addresses the problem of estimating the state sequence of a Markov process observed through noise using the trellis based decoding algorithms. Section 3.4 presents the Viterbi Algorithm and its modified version to yield soft output decoding algorithm that deliver soft outputs. This algorithm is known as Soft Output Viterbi Algorithm (SOVA). Section 3.5 summarizes the chapter.

3.2 *Iterative Decoding Principles*

The problem of decoding turbo codes involves the joint estimation of two Markov processes, one for each constitute encoder. Present theory of turbo codes uses a single Markov process model. Turbo decoders use two trellis-based soft-input/soft-output decoding algorithms. The two Markov processes are linked by an interleaver. Additional coding gain can be achieved by sharing information between the two decoders in an iterative fashion. This is achieved by allowing the output of one decoder to be used as a priori information by the other decoder.

3.2.1 *Log-Likelihood Algebra*

To best explain the iterative feedback of soft decisions, the idea of log-likelihood algebra is used [56]. Let U be a binary random variable in the GF(2) with elements {+1,-1}, where +1 is the null element under the $\oplus$ addition. The log-likelihood ratio of a binary random variable U is $L(u)$ and it is given as following:

$$L(u) = \log \frac{P(u=+1)}{P(u=-1)} \tag{3-1}$$

where $P(u)$ denotes the probability that the random variable U takes the value u. Unless otherwise stated, the logarithm is the natural logarithm. $L(u)$ denotes the soft value or L-value of a binary random variable. The sign of $L(u)$ corresponds to the hard decision, while the magnitude, $|L(u)|$, gives the reliability of the decision.

If the binary random variable X is conditioned on a different random variable Y, then we have a conditional log-likelihood ratio $L(x/y)$ defined as follows:

$$L(x|y) = \log \frac{P(x=+1|y)}{P(x=-1|y)}$$

$$= \log \frac{P(x=+1)}{P(x=-1)} + \log \frac{P(y|x=+1)}{P(y|x=-1)}$$

$$= L(x) + L(y \mid x) \tag{3-2}$$

For completeness, the three different types of addition used in this analysis are described. Ordinary addition is defined by the plus sign (+), modulo-2 addition is defined by the circled plus sign ($\oplus$). Log-likelihood addition is defined by the bracketed plus sign [+] [49]. The sum of two log-likelihood ratios (LLR) is denoted by the operator [+], where such an addition is defined as the LLR of modulo-2 sum of the underlying data bits. This addition can be expressed mathematically as following:

$$L(x_1) \,[+]\, L(x_2) \stackrel{\Delta}{=} L(x_1 \oplus x_2) \tag{3-3}$$

And $L(x_1 \oplus x_2)$ is given as following:

$$L(x_1 \oplus x_2) = \log \frac{P(x_1 \oplus x_2 = +1)}{P(x_1 \oplus x_2 = -1)}$$

$$= \log \frac{P(x_1 \oplus x_2 = +1)}{1 - P(x_1 \oplus x_2 = +1)} \tag{3-4}$$

For statistically independent random variables x_1 and x_2, $P(x_1 \oplus x_2=+1)$ is given by the following:

$$P(x_1 \oplus x_2) = P(x_1 = +1)P(x_2 = +1) + P(x_1 = -1)P(x_2 = -1)$$

$$= P(x_1 = +1)P(x_2 = +1) + [1 - P(x_1 = +1)][1 - P(x_2 = +1)] \qquad (3\text{-}5)$$

Using the definition of $L(x)$ given in Equation 3-1, $P(x)$ is obtained by taking the exponential of both sides:

$$e^{L(x)} = \frac{P(x = +1)}{P(x = -1)} = \frac{P(x = +1)}{1 - P(x = +1)} \qquad (3\text{-}6)$$

Then, $$P(x = +1) = \frac{e^{L(x)}}{1 + e^{L(x)}} \qquad (3\text{-}7)$$

Using Equations 3-5 and 3-7 and substituting into Equation 3-4:

$$L(x_1 \oplus x_2) = \log \frac{P(x_1 \oplus x_2 = +1)}{P(x_1 \oplus x_2 = -1)} = \log \frac{P(x_1 \oplus x_2 = +1)}{1 - P(x_1 \oplus x_2 = +1)} \qquad (3\text{-}8)$$

Substituting the results of Equation 3-5 into Equation 3-8 yields:

$$L(x_1 \oplus x_2) = \log \frac{P(x_1 = +1)P(x_2 = +1) + [1 - P(x_1 = +1)][1 - P(x_2 = +1)]}{[1 - P(x_1 = +1)P(x_2 = +1)] - [1 - P(x_1 = +1)P(x_2 = +1)]} \qquad (3\text{-}9)$$

Using the results from Equation 3-7 to substitute into Equation 3-9. The result can then be multiplied by $\frac{(1+e^{L(x_1)})(1+e^{L(x_2)})}{(1+e^{L(x_1)})(1+e^{L(x_2)})}$ to produce:

$$L(x_1 \oplus x_2) = \log \frac{1+e^{L(x_1)}e^{L(x_2)}}{e^{L(x_1)}+e^{L(x_2)}} \tag{3-10}$$

The right hand side of Equation 3-10 can approximated as in [56] by:

$$L(x_1 \oplus x_2) \approx sign(L(x_1))\ \ sign(L(x_2))\ \ \min(|L(x_1)|,|L(x_2)|) \tag{3-11}$$

The approximation in Equation 3-11 yields the following interesting results when one of the LLRs is very large or very small:

$$L(x)\ [+]\ \square = L(x) \tag{3-12}$$

$$L(x)\ [+]\ -\square = -L(x) \tag{3-13}$$

and, $$L(x)\ [+]\ 0=0 \tag{3-14}$$

By induction, for any integer number J, further results can be proved as following:

$$\sum_{j=1}^{J}[+]L(x_j) \overset{\Delta}{=} L(\sum_{j=1}^{J} \oplus x_j) \tag{3-15}$$

Using the approximation in Equation 3-11, Equation 3-15 can be written as:

$$\sum_{j=1}^{J}[+]L(x_j) \approx \left(\prod_{j=1}^{J} sign(L(x_j))\right) \min_j |L(x_j)| \tag{3-16}$$

3.2.2 Soft Channel Outputs

If a set of binary bits x is transmitted with soft values $L(x)$ over an additive white Gaussian noise (AWGN) channel with fading, the log-likelihood ratio of x conditioned on the matched filter output y can be shown as:

$$L(x|y) = \log \frac{P(x=+1|y)}{P(x=-1|y)}$$

$$= \log \frac{P(y|x=+1)}{P(y|x=-1)} + \log \frac{P(x=+1)}{P(x=-1)} \tag{3-17}$$

In the case of an additive white Gaussian channel with fading, the probability density function of y given x with a fading depth a is:

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-ax)^2}{2\sigma^2}} \tag{3-18}$$

where $\sigma^2 = \frac{N_0}{2E_s}$ is the noise variance. Using Equation 3-18 further simplification of Equation 3-17 can be achieved:

$$L(x|y) = L_c y + L(x) \tag{3-19}$$

where $L_c = \frac{4E_s}{N_0} a$ is the reliability of the channel. For Gaussian channel, $a = 1$.

3.2.3 *Iterative Decoding*

Having the L-values, a priori and soft channel values, the iterative decoding of turbo codes can be formulated, Figure 3-1 illustrates a half iteration of turbo code decoder which is represented by a soft-input/soft-output decoder.

The soft-input/soft-output decoder accepts a priori values $L(m)$ for all information bits m, and channel values $L_c y$ for all coded bits. It delivers soft outputs $L(\hat{m})$ on all information bits and an extrinsic information $L_e(\hat{m})$ which contains the soft output information from all the other coded bits in the code sequence. The extrinsic information is not influenced by the $L(m)$ and $L_c y$ values of the current bit.

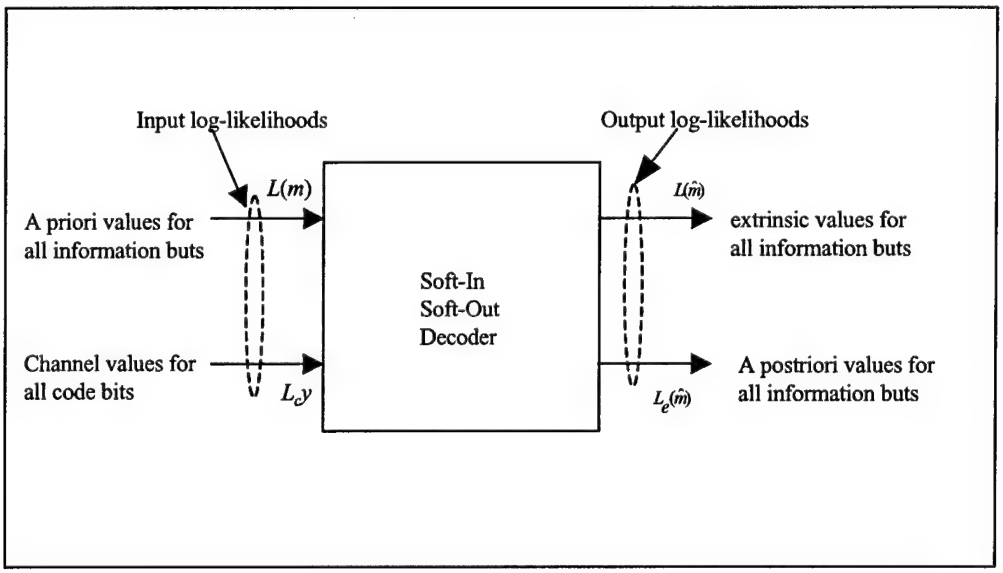

Figure 3-1 Soft-Input/Soft-Output Decoder [56].

For systematic codes, the soft output for the information bit m is represented by three additive terms as follows:

$$L(\hat{m}) = L_c y + L(m) + L_e(\hat{m}) \tag{3-20}$$

This means that there are three independent estimates for the log-likelihood ratio of the information bits: the channel values $L_c y$, the a priori values $L(m)$, and the extrinsic values of $L_e(\hat{m})$. Assuming no a priori information is available for the first iteration, $L(m)$ is initialized to zero. Decoding of the first component of the turbo code decoder starts by using the corresponding $L_c y$. The first component decoder can be denoted by the horizontal decoder, C^{-}. The extrinsic information, $L^{-}{}_e(\hat{m})$,obtained from the horizontal code decoder C^{-} for the information bit m is given by:

$$L^{-}{}_e(\hat{m}) = L^{-}(m) - L_c y \tag{3-21}$$

This independent estimate on m is now used as the a priori values for decoding the second component decoder. This second component can be denoted by the vertical decoder $C^{|}$. After the second half of the iteration, the vertical extrinsic information is:

$$L^{|}_e(\hat{m}) = L^{|}(m) - L_c y - L^{-}_e(\hat{m}) \tag{3-22}$$

This vertical extrinsic information is used as a new a priori value in the subsequent decoding of code C^{-} in the next iteration step. This process is shown in Figure 3-2. Note that for the first horizontal and the first vertical iteration, the L-values are statistically independent. Later iterations use the same information indirectly and as such become more and more correlated. As the number of iterations increase, the improvement

becomes marginal. Following the last vertical iteration, the final decision is obtained by combining the last two extrinsic pieces of information with the received values. This yields:

$$L(\hat{m}) = L_c y + L_e^{-}(\hat{m}) + L_e^{|}(\hat{m}) \tag{3-23}$$

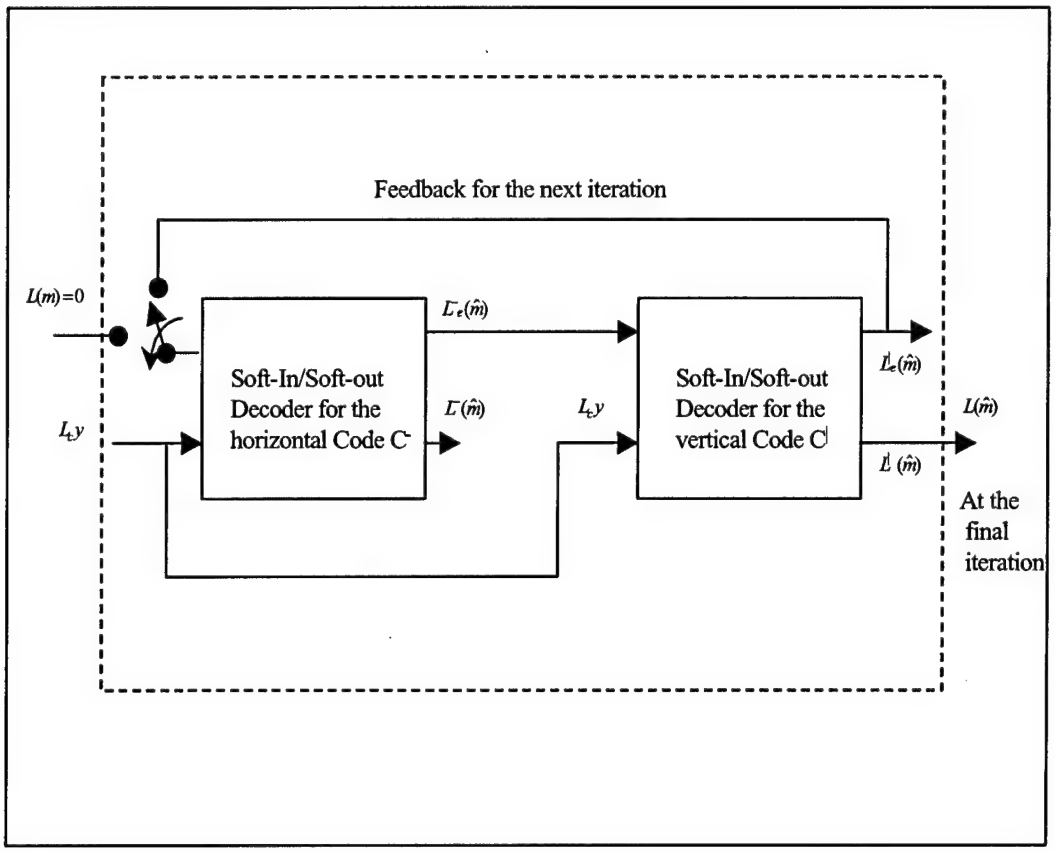

Figure 3-2 Iterative decoding procedure with two "soft-in/soft-out" decoders [56].

3.3 Trellis Based Decoding

A binary convolutional encoder with rate $1/N$ and constraint length K (number of shift registers) is a finite memory system that outputs N binary digits for each information digit presented at its input. Figure 3-3 shows a typical convolutional encoder [11]. The state of the convolutional encoder is determined by the contents of the right most (K-1) elements of the shift register. There are then 2^{K-1} possible states for the encoder. Given possible the encoder receives a new input, it can only make a transition to one of the two other states, depending on whether the input bit is "1" or "0". When the encoder makes a transition from one state to another, a particular sequence of N bits is output. When the input bits are grouped in frames of length L, the encoder makes L transitions to produce a sequence of LN output bits.

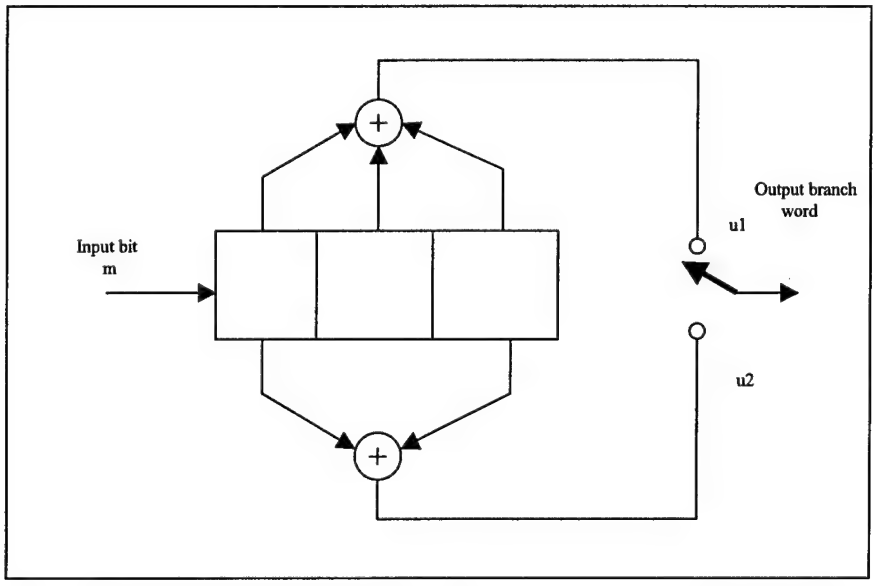

Figure 3-3 Convolutional encoder (rate1/2, $K = 3$) [11].

An alternative approach can be taken to produce an algebraic description of the convolutional encoder. This alternative is based on the observation that the convolutional encoder is a finite memory system. Its output sequence depends on the input sequence and the state of its shift registers. One description is obtained from the state diagram of the convolutional encoder. This representation is shown in Figure 3-4 for the encoder in Figure 3-3. The states, shown in the boxes of the diagram, represent possible contents of the right most (K-1) shift registers. The paths between the states represent the output branch words resulting from such state transitions. The register states are designated a = 00, b = 10, c = 01, and d = 11. Figure 3-4 illustrates all state transitions that are possible for the encoder in Figure 3-3.

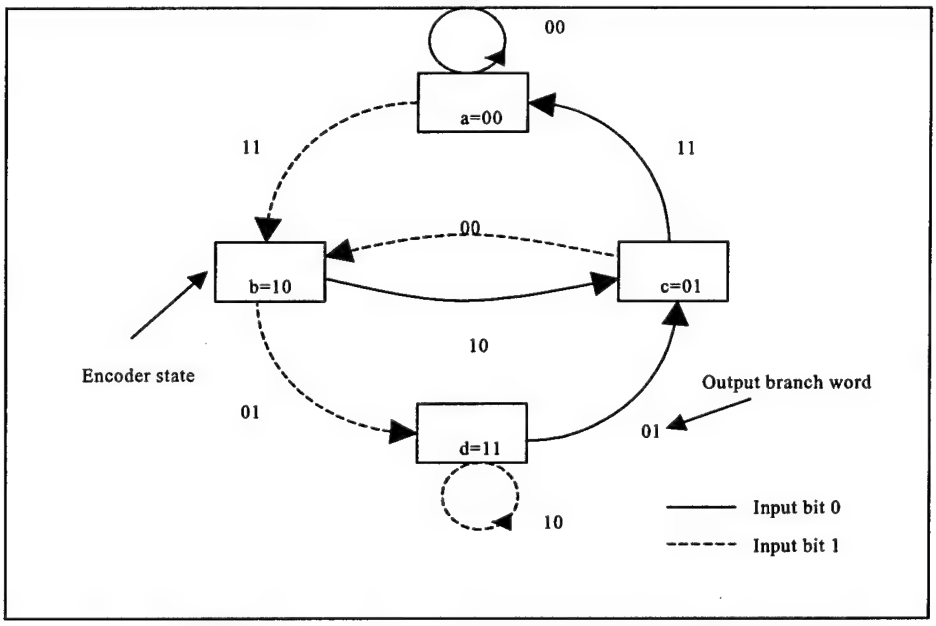

Figure 3-4 Encoder state diagram (rate 1/2, K=3).

There are only two possible transitions from each state, corresponding to two possible input bits. Next to each path connecting states is written the output branch word associated with the state transition. The following example [11] illustrates the transition sequence of the convolutional encoder as if responds to an arbitrary input sequence.

Example 3-1: For the encoder shown in Figure 3-3, show the state changes and resulting output codeword sequence U for the message m = 11011, followed by K-1 = 2 zeros to flush the register. Assume the initial contents of the shift registers are zeros.

Solution: The output branch words are shown below in Table 3-1 for the message input m = *1101100.*

Table 3-1. Output branch words according to the input bits and the contents of the registers [11].

Input bit m_i	*Register contents*	*State at time i*	*State at time i+1*	*Branch word at time i*	
				u_1	u_2
-	0 0 0	0 0	0 0	-	
1	1 0 0	0 0	1 0	1	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1

The output sequence: U = 1 1 0 1 0 1 0 0 0 1 0 1 1 1

Although the state diagram completely characterizes the encoder, one cannot easily use it for tracking the encoder transitions as a function of time since the diagram cannot represent time history. By adding the dimension of time to the state diagram, the dynamic description of the encoder as a function of a particular input sequence can be achieved. The state diagram can be expanded into a trellis diagram which explicitly shows the passage of time. The trellis diagram for the convolutional encoder of Figure 3-3 is shown in Figure 3-5. In the trellis diagram, as in the state diagram, the solid lines are denote the output generated by an input bit zero and dashed lines denote the output generated by an input bit one. The trellis nodes characterize the encoder states. The first row nodes correspond to the state $a = 00$, while the second and subsequent rows correspond to the states $b = 10$, $c = 01$, and $d = 11$. At each unit of time, the trellis requires 2^{K-1} nodes to represent the 2^{K-1} possible encoder states. The output branch words corresponding to the state transitions appear as labels on the trellis branches in Figure 3-5. At the receiving end, the decoder needs to estimate the state sequence of the convolutional encoder output observed through noise.

Problem Statement: *Given a sequence Y, representing observations of a discrete-time finite-state Markov process in memoryless noise, estimate the state sequence of the Markov process.*

There are two well-known trellis-based solutions to this problem. One is called the Viterbi Algorithm [46] and is based on minimizing the sequence error probability. The other one is called the *Maximum a Posteriori* (MAP) algorithm and is based on

minimizing the bit error probability (symbol-by-symbol) [47]. The MAP algorithm attempts to maximize the a posteriori probabilities (APP) of each individual bit in the sequence. Until recently, this algorithm received little attention because its computational complexity exceeds the Viterbi algorithm and provides little advantage in bit error rate performance.

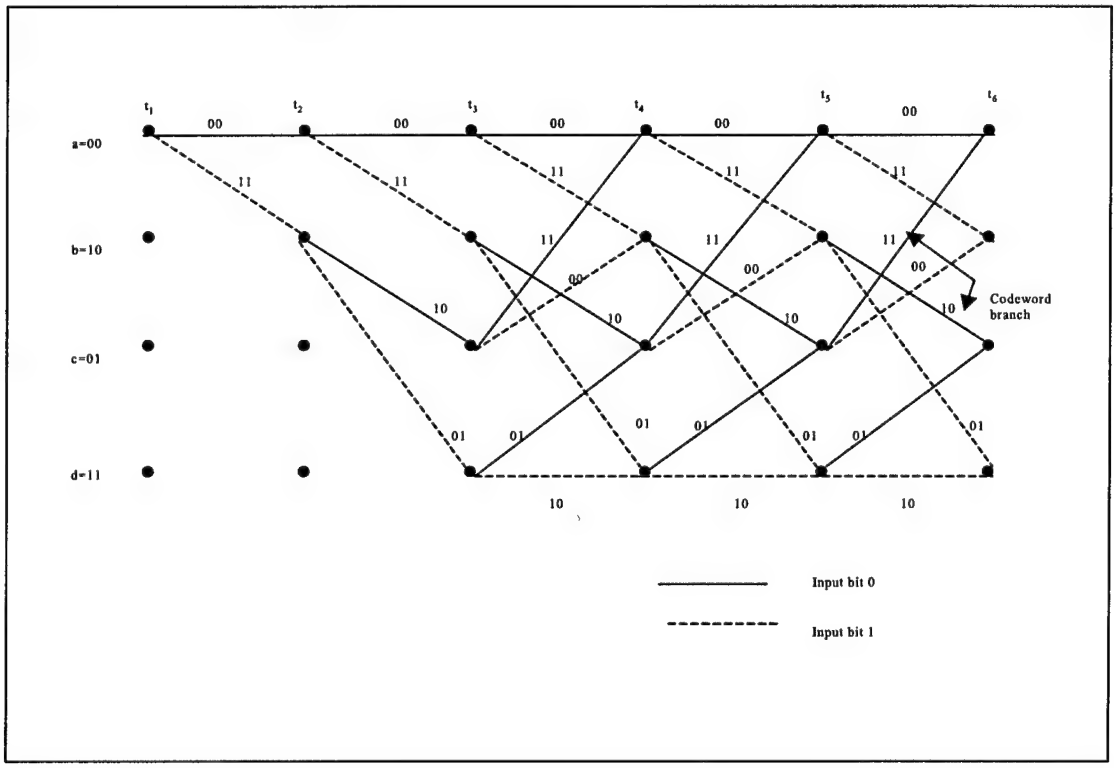

Figure 3-5 Encoder trellis diagram (rate 1/2 , $K = 3$).

The Viterbi algorithm performs the maximum likelihood estimation of the transmitted sequence. Its output represents the code sequence closest (Hamming or

Euclidean distance) to the received one. The MAP algorithm differs from maximum likelihood decoding in that it does not assume equally likely information symbols. The big difference between the Viterbi and MAP algorithms lie in their output. The Viterbi algorithm outputs a *hard* decision of transmitted digits, whereas, the MAP algorithm provides a posteriori probabilities which may be interpreted as a *soft* estimate of transmitted digits. When convolutional codes are employed in concatenating codes, e.g., turbo coding, this difference becomes fundamental and explains the great revival of interest in MAP algorithm implementation.

Turbo coding research has used both algorithms for the constituent decoder in the iterative stages of turbo code decoding. One direction lets the Viterbi algorithm provide a soft output for each symbol and is known as the Soft-Output Viterbi Algorithm (SOVA) [50-52]. The other direction simplifies the complexity of the MAP algorithm for implantation [53-57].

3.4 Viterbi Algorithm with Soft decision

The Viterbi Algorithm (VA) was originally proposed in 1967 for decoding convolutional codes [85]. This algorithm is a recursive optimal solution to the problem of estimating the state sequence of a discrete-time finite state Markov process observed in memoryless noise. In 1969, Omura [86] demonstrated that the Viterbi algorithm provides a maximum likelihood solution.

The original VA produced only hard decisions and is not suitable for turbo code decoder implementation. A modified VA is used to deliver the most likely path sequence and the either a posteriori probability for each bit or a reliability value. With the

reliability indicator, the modified VA produces soft decisions for use by subsequent decoding and is called the Soft-output Viterbi Algorithm (SOVA). This algorithm *accepts* and *delivers* soft sample values.

3.4.1 Viterbi Algorithm (VA)

In its most general form, the Viterbi algorithm may be viewed as a solution to the problem of *maximum a posteriori probability (MAP) estimation* of the state sequence. This solution is defined in a finite-state discrete-time Markov process as observed in memoryless noise.

Let $S^{(m)} = (s_0^{(m)}, \ldots, s_L^{(m)})$ denote the state sequence beginning at time t_o and terminating at time t_L, the *mth* codeword. Given observation sequence Y of a discrete-time finite-state Markov process in memory less noise, the VA finds the state sequence, $S^{(m)}$, for which the a posteriori probability $P(S^{(m)} \mid Y)$ is maximum.

Let $S^{(m')}$ denote the state sequence solution, written as:

$$S^{(m')} = \arg\left\{\max_m \; P(S^{(m)}|Y)\right\} \tag{3-24}$$

Using Bayes theorem,

$$S^{(m')} = \arg\left\{\max_m \; \frac{P(Y|S^{(m)})\; P(S^{(m)})}{P(Y)}\right\} \tag{3-25}$$

Noting that $P(Y)$ is the same for all state sequence estimations, then:

$$S^{(m')} = \arg\left\{\max_{m} \; P(Y|S^{(m)}) \; P(S^{(m)})\right\} \tag{3-26}$$

$$S^{(m')} = \arg\left\{\max_{m} \; P(S^{(m)}, Y)\right\} \tag{3-27}$$

For L length sequences, a brute force method could calculate the joint probability $P(S^{(m)}, Y)$ for all possible state sequences m (codewords or paths), the state sequence (path) with the largest value would then be selected and information bits corresponding to that state sequence (path) would form the decoder output. Unfortunately, the number of paths for an L-bit information sequence is 2^L. Thus, the brute force decoding approach quickly becomes computationally impractical as L increases.

The Viterbi algorithm uses the advantage of the underlying trellis structure to greatly reduce the required number of computations. Since the process is Markovian, the probability that the process is in a particular state, given all past states, only depends on the previous state:

$$P(s_{i+1}^{(m)} | s_0^{(m)}, \ldots, s_i^{(m)}) = P(s_{i+1}^{(m)} | s_i^{(m)}) \tag{3-28}$$

The probability of a particular observation, y_i, given the entire state sequence, $S^{(m)}$, is the probability of the observation given the previous state transition $s_i^{(m)} \rightarrow s_{i+1}^{(m)}$:

$$P(y_i | S^{(m)}) = P(y_i | s_i^{(m)} \rightarrow s_{i+1}^{(m)}) \tag{3-29}$$

Assuming the channel affects channel symbols independently, the $P(S^{(m)}, Y)$ given in Equation 3-27 can be expressed as:

$$P(S^{(m)},Y)=\prod_{i=0}^{L-1}P(y_i\left|s_i^{(m)}\rightarrow s_{i+1}^{(m)}\right.)\prod_{i=0}^{L-1}P(s_{i+1}^{(m)}\left|s_i^{(m)}\right.) \tag{3-30}$$

To avoid multiplication (i.e., simplify computations), the logarithm of the multiplicand values provides a preferable path metric given by:

$$M^{(m)}=\log\ P(S^{(m)},Y)$$

$$=\sum_{i=0}^{L-1}\log P(y_i\left|s_i^{(m)}\rightarrow s_{i+1}^{(m)}\right.)+\sum_{i=0}^{L-1}\log P(s_{i+1}^{(m)}\left|s_i^{(m)}\right.) \tag{3-31}$$

where $M^{(m)}$ is the accumulated metric of state sequence m. In most cases, the natural logarithm is used. If we denote the branch metric associated with the transition $s_i^{(m)}\rightarrow s_{i+1}^{(m)}$ by $\Gamma(s_i^{(m)}\rightarrow s_{i+1}^{(m)})$, then

$$\Gamma(s_i^{(m)}\rightarrow s_{i+1}^{(m)})=\log P(y_i\left|s_{i+1}^{(m)}\right.)+\log P(s_{i+1}^{(m)}\left|s_i^{(m)}\right.) \tag{3-32}$$

For a rate *1/N* convolutional code, the observation $y_i=(y_{i,0},\ldots,y_{i,N-1})$ is a noisy (and possibly faded) observation of the encoder output (or the channel input) $x_i^{(m)}=(x_{i,0},\ldots,x_{i,N-1})$ of the Markov process, and not a direct observation of the states. Also, for every state transition $s_i^{(m)}\rightarrow s_{i+1}^{(m)}$, there is a message m_i producing the state transition. There is also a one-to-one correspondence between both of them. Thus, it is more convenient to write the branch metric in terms of transmitted symbols and underlying message rather than the state sequence. Using this fact, Equation 3-32 can be rewritten as:

$$\Gamma(s_i^{(m)}\rightarrow s_{i+1}^{(m)})=\log P(y_i\left|x_i^{(m)}\right.)+\log P(m_i^{(m)}) \tag{3-33}$$

For a rate *1/N* convolutional code, $\Gamma(s_i^{(m)} \rightarrow s_{i+1}^{(m)})$ in Equation 3-33 can be written as:

$$\Gamma(s_i^{(m)} \rightarrow s_{i+1}^{(m)}) = \sum_{n=0}^{N-1} \log P(y_{i,n} \left| x_{i,n}^{(m)} \right.) + \log P(m_i^{(m)}) \tag{3-34}$$

where $x_{i,n}^{(m)}$ is the *nth* coded symbol(s) corresponding to information m_i in the m message sequence. If message bits are assumed to be equiprobable, the term $P(m_i^{(m)})$ is the same for all hypotheses and can be omitted from Equation 3-34. The Viterbi algorithm only calculates $P(y_{i,n} \left| x_{i,n}^{(m)} \right.)$ terms in its search. For this case, the Viterbi algorithm is said to be a *maximum likelihood estimator,* and the branch metric is:

$$\Gamma(s_i^{(m)} \rightarrow s_{i+1}^{(m)}) = \sum_{n=0}^{N-1} \log P(y_{i,n} \left| x_{i,n}^{(m)} \right.) \tag{3-35}$$

Using the channel transition probabilities $P(y_{i,n} \left| x_{i,n}^{(m)} \right.)$ only, the VA calculates the accumulated branch metric, $M^{(m)}$, for all possible paths using the following equation:

$$M^{(m)} = \sum_{i=0}^{L-1} \Gamma(s_i^{(m)} \rightarrow s_{i+1}^{(m)})$$

$$= \sum_{i=0}^{L-1} \sum_{n=0}^{N-1} \log P(y_{i,n} \left| x_{i,n}^{(m)} \right.) \tag{3-36}$$

Substituting Equation 3-36 to Equation 3-27, yilds:

$$S^{(m')} = \arg\left\{ \max_{m} \; M^{(m)} \right\}$$

$$= \arg\left\{ \max_{m} \sum_{i=0}^{L-1} \sum_{n=0}^{N-1} \log P(y_{i,n} | x_{i,n}^{(m)}) \right\} \tag{3-37}$$

The Viterbi algorithm calculates a measure of similarity, or distance, between the received signal at time i and all trellis paths entering each state at time i. The VA removes from consideration those trellis paths that can not possibly be candidates for the maximum likelihood choice. When two paths enter the same state, the one having the best metric is chosen and is called the surviving path. The selection of surviving paths is performed for all states. The decoder continues in this way, advancing deeper into the trellis and making decisions by eliminating least likely paths. The early rejection of unlikely paths reduces the decoding complexity. Since the channel transition probabilities $P(y_{i,n} | x_{i,n}^{(m)})$ only depend on paths under consideration.

A Binary Systematic Channel (BSC) is a discrete memoryless channel that has binary input and output alphabets and symmetric transition probabilities. It can be described by the following conditional probabilities:

$$P(0|1) = P(1|0) = p \tag{3-38}$$

$$P(1|1) = P(0|0) = 1 - p \tag{3-39}$$

where p is the probability that the output symbol from the channel is different from the input symbol. Let $X^{(m)}$ be a transmitted codeword over a BSC with symbol error probability p, and let Y be the corresponding received word. Suppose that $X^{(m)}$ and Y are each NL-bit length sequences and that they differ in d_m positions (i.e., the Hamming

distance between them is d_m). Since the channel is memoryless, the probability that this $X^{(m)}$ was transformed to the specific received word Y at distance d_m can be written as:

$$P\left(Y\middle|X^{(m)}\right)=\prod_{i=0}^{L-1}p(y_i\middle|x_i^{(m)}) \tag{3-40}$$

As mentioned before if the output symbols of the channel, Y, are different from a specific code word $X^{(m)}$ in d_m positions, then

$$P\left(Y\middle|X^{(m)}\right)=p^{d_m}(1-p)^{L-d_m} \tag{3-41}$$

From Equation 3-41, the log-likelihood function is:

$$\log P(Y\middle|X^{(m)}) = d_m \log p + (L-d_m)\log(1-p)$$

$$= -d_m \log(\frac{1-p}{p}) + L\log(1-p) \tag{3-42}$$

When computing this quantity for each possible transmitted sequence, $X^{(m)}$, the second term is constant in each case. Assuming that $p < 0.5$, we can express Equation 3-42 as following:

$$\log P(Y\middle|X^{(m)}) = -A\ d_m - B \tag{3-43}$$

where A and B are positive constants. Substituting the log-likelihood of the received codeword in Equation 3-43 into Equation 3-37. Accordingly, maximum likelihood decoding chooses codeword $X^{(m)}$, or message m, that maximizes the log-likelihood function which in-turn minimizes the Hamming distance, d_m.

For a faded Gaussian channel with Binary Phase Shift Key (BPSK) modulation, the output of the channel (demodulator) corresponding to the *ith* bit may be expressed as:

$$y_{i,n} = a_{i,n}\ x_{i,n}\sqrt{E_s} + n'_{i,n} \tag{3-44}$$

where $n'_{i,n}$ denotes Additive White Gaussian Noise at the receiver output with variance $N_0/2$. The parameter $x_{i,n}$ denotes the transmitted symbol taking on values of ± 1. The fading depth is denoted by $a_{i,n}$ while $y_{i,n}$ is a Gaussian random variable with mean $a_{i,n}x_{i,n}\sqrt{E_s}$. The energy per symbol for this relationshipo is denoted by E_s. This relationship assumes perfect side information (i.e., at each symbol the fading depth $a_{i,n}$ is known) is provided to the decoder. When $a_{i,n} = 1$, the channel is said to be an Additive White Gaussian Noise (AWGN) channel, otherwise the channel is said to be faded. An alternate and more convenient expression for Equation 3-44 is:

$$y_{i,n} = a_{i,n}\ x_{i,n} + n_{i,n} \tag{3-45}$$

where the noise variance now $\sigma^2_{i,n} = \frac{N_0}{2E_s}$ and the probability density function of $y_{i,n}$ given $x_{i,n}$ and $a_{i,n}$ is:

$$p(y_{i,n}|x_{i,n}, a_{i,n}) = \frac{1}{\sqrt{2\pi\sigma^2_{i,n}}} e^{\frac{(y_{i,n}-a_{i,n}x_{i,n})^2}{2\sigma^2_{i,n}}} \tag{3-46}$$

Since each symbol is affected independently by the AWGN, $\sigma^2_{i,n} = \sigma^2$. The likelihood function of the path sequence, *m*, can be expressed as:

$$p(y|X^{(m)},a)=\prod_{i=0}^{L-1}\prod_{n=0}^{N-1}\frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{(y_{i,n}-a_{i,n}x_{i,n}^{(m)})^2}{2\sigma^2}} \tag{3-47}$$

Taking the natural logarithm of both sides of Equation 3-47, the log-likelihood function of $p(y|X^{(m)},a)$ becomes:

$$\ln\ p(y|X^{(m)},a)=\sum_{i=0}^{L-1}\sum_{n=0}^{N-1}\ln\frac{1}{\sqrt{2\pi\sigma^2}}-\frac{\left(y_{i,n}-a_{i,n}x_{i,n}^{(m)}\right)^2}{2\sigma^2} \tag{3-48}$$

Eliminating all components that are independent of message, m, and substituting into Equation 3-37:

$$S^{(m')}=\arg\left\{\max_{m}\sum_{i=0}^{L-1}\sum_{n=0}^{N-1}(y_{i,n}-a_{i,n}x_{i,n}^{(m)})^2\right\} \tag{3-49}$$

From Equation 3-49, the branch metric in this case becomes:

$$\Gamma\left(s_i^{(m)}\rightarrow s_{i+1}^{(m)}\right)=(y_{i,n}-a_{i,n}x_{i,n}^{(m)})^2 \tag{3-50}$$

which can be further simplified to:

$$(y_{i,n}-a_{i,n}x_{i,n}^{(m)})^2=y_{i,n}^2-2a_{i,n}x_{i,n}^{(m)}y_{i,n}+(a_{i,n}x_{i,n}^{(m)})^2 \tag{3-51}$$

Note that $y_{i,n}^2$ is independent of any specific message, m, and $\left(a_{i,n}x_{i,n}^{(m)}\right)^2$ is the same for all transitions (or branches) occuring at the same time. Therefore, the branch metric can be rewritten as:

$$\Gamma\left(s_i\rightarrow s_{i+1}\right)=-a_{i,n}x_{i,n}^{(m)}y_{i,n} \tag{3-52}$$

By maximizing the accumulated branch metric, $M^{(m)}$, the maximum likelihood codeword (or message) can be chosen according to the following equation:

$$S^{(m')} = \arg\left\{ \max_{m} \sum_{i=0}^{L-1} \sum_{n=0}^{N-1} - a_{i,n} x_{i,n}^{(m)} y_{i,n} \right\}$$

$$= \arg\left\{ \min_{m} \sum_{i=0}^{L-1} \sum_{n=0}^{N-1} a_{i,n} x_{i,n}^{(m)} y_{i,n} \right\} \qquad (3\text{-}53)$$

This is equivalent to choosing the codeword (or message) that is closest in Euclidean distance to Y. Even though the hard and soft decision channels require different metrics, the concept of choosing the message m' that is closest to the received sequence, Y, is the same in both cases. To implement the maximization of Equation 3-37 exactly, the decoder has to be able to handle analog valued arithmetic operations. This is impractical because the decoder is generally implemented digitally. Thus, it is necessary to quantize the received symbols, $y_{i,n}$.

To understand Viterbi's decoding algorithm, it is convenient to illustrate the algorithm by an example [11]. The encoder for this example is shown in Figure 3-3, and the encoder trellis diagram is shown in Figure 3-5. A similar trellis can be used to represent the decoder as shown in Figure 3-6. For simplicity, a Binary Systematic Channel (BSC) is assumed, and thus the Hamming distance is a proper distance measure. For the decoder trellis shown in Figure 3-6, it is convenient to label each trellis branch at time i with the Hamming distance between the received code symbols and the corresponding branch word from the encoder trellis. The example in Figure 3-6 shows a message m = 11011. The corresponding codeword sequence U = 11 01 01 00 01 and a

noise corrupted received modulated sequence, Y = 11 01 01 10 01. The branch words seen on the encoder trellis branches characterize the encoder in Figure 3-3 and are known a priori to the both the encoder and the decoder. These encoder branch words are the code symbols that would be expected to come from the encoder output as a result of each of the transitions. The labels on the decoder trellis branches are accumulated by the decoder as they are calculated. That is, as the code symbols are received, each branch of the decoder trellis is labeled with a metric of similarity (Hamming distance) between the received code symbols and each of the branch words for that time interval. From the received sequence, Y, in Figure 3-6, the code symbols received at time i = 1 are 11. In order to label the decoder branches at time i = 1 with the appropriate Hamming distance metric, Figure 3-5 is used. Here, a state $00 \rightarrow 00$ transition yields an output branch word of 00, but a 1 was received. Therefore, on the decoder trellis the state $00 \rightarrow 00$ transition is labeled with the Hamming distance of two. The decoder trellis branches are labeled in this way as the symbols are received at each time. The decoding algorithm uses these Hamming distance metrics to find the most likely (minimum distance) path through the trellis. The cumulative Hamming path metric of a given path at time i is the sum of the branch Hamming distance metric along that path up to time i.

In the Viterbi decoding, if any two paths in the trellis merge to a single state, one of them can be always be eliminated in the search for an optimum path. For example, Figure 3-7 shows two paths merging at time i = 5 to state 00, the upper path has metric 4 the lower has metric 1. The upper path cannot be a portion of the optimum path because the lower path, which enters the same state summarizes the encoder history in the sense that the previous states cannot affect the future states or future output branches.

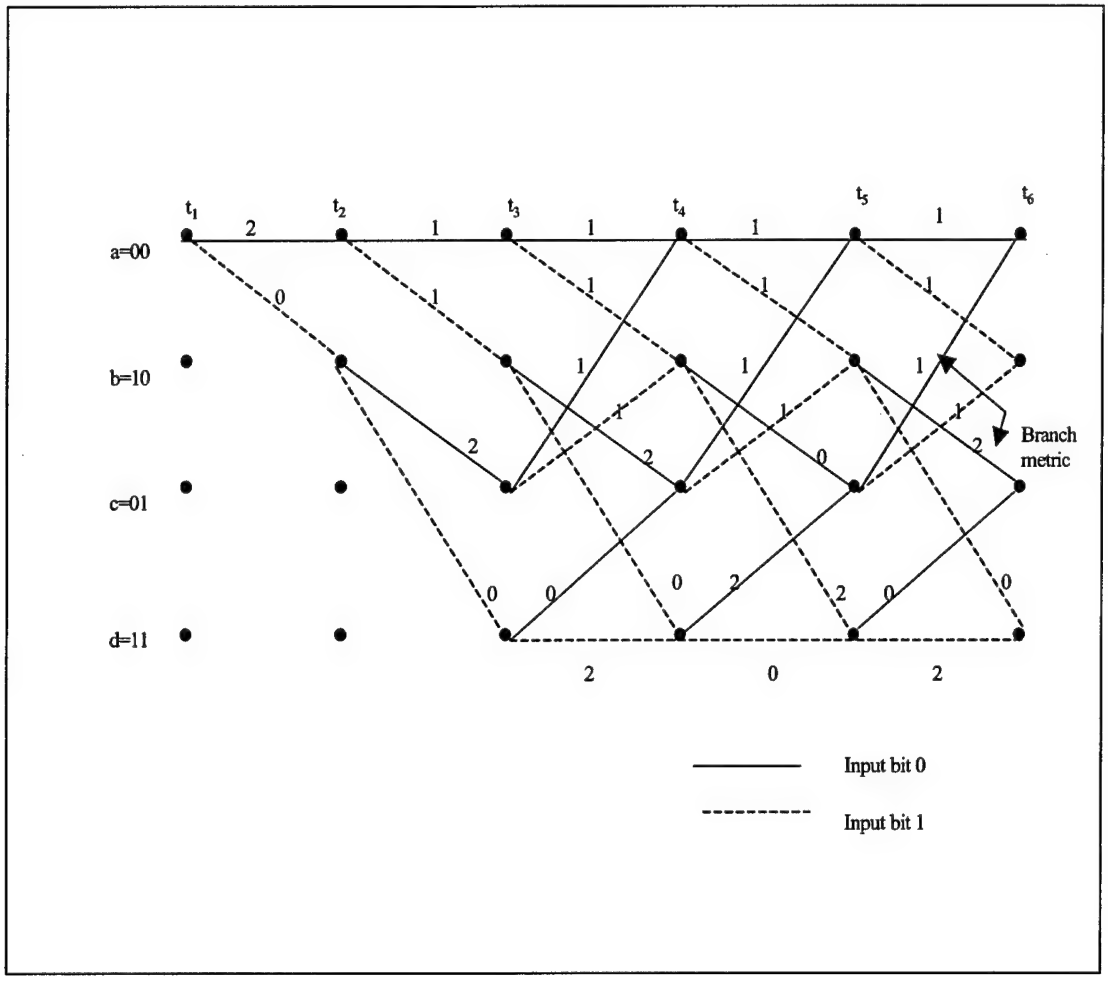

Figure 3-6 Decoder trellis diagram (rate 1/2, K=3) [11].

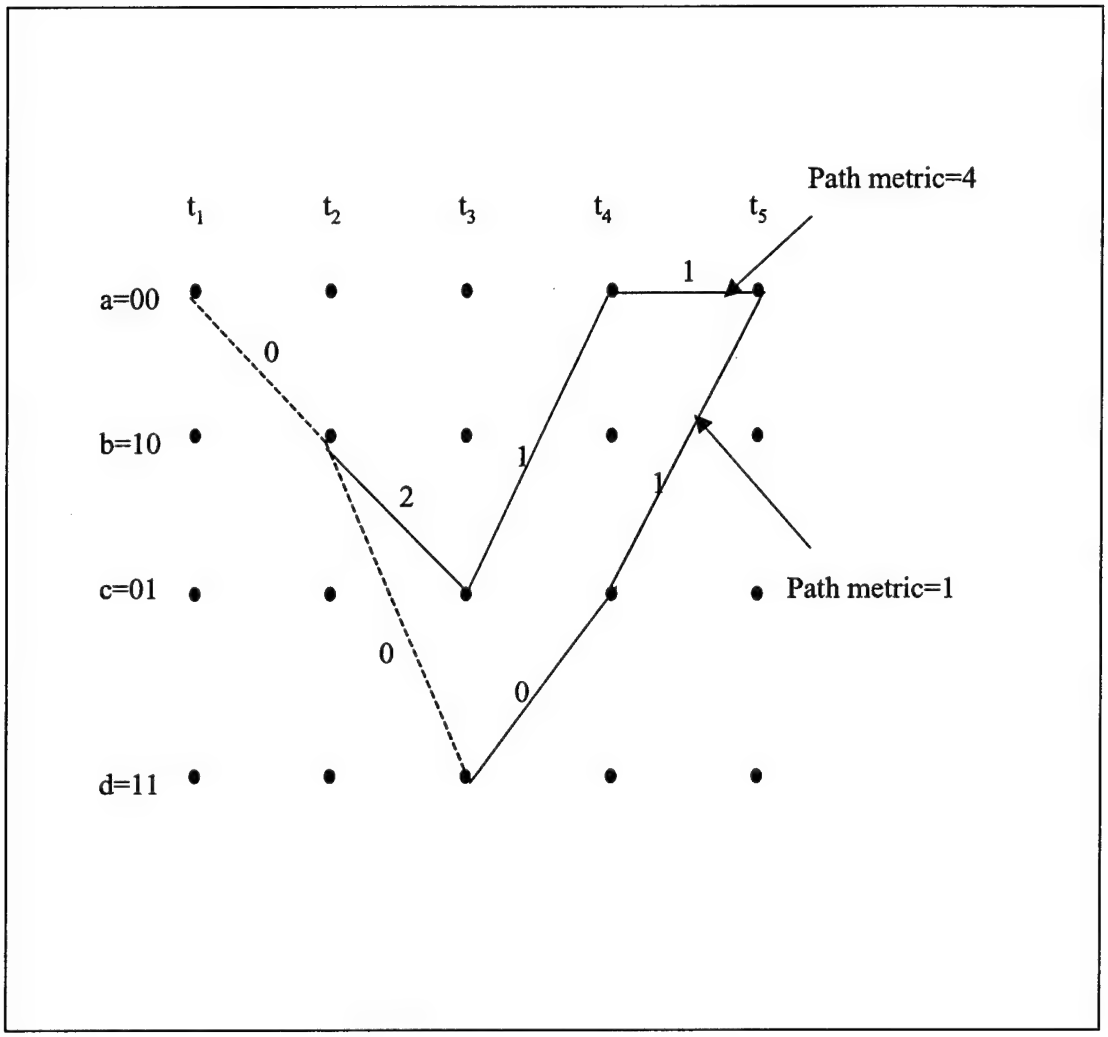

Figure 3-7 Path metric for two merging paths [11].

The first few steps in this decoding example are shown in Figure 3-8. In Figure 3-8, M_a, M_b, M_c, and M_d, represent the cumulative metrics of the paths that terminate at the corresponding states. Assume the input data sequence m, codeword U, and the received sequence Y are given as in the previous example.

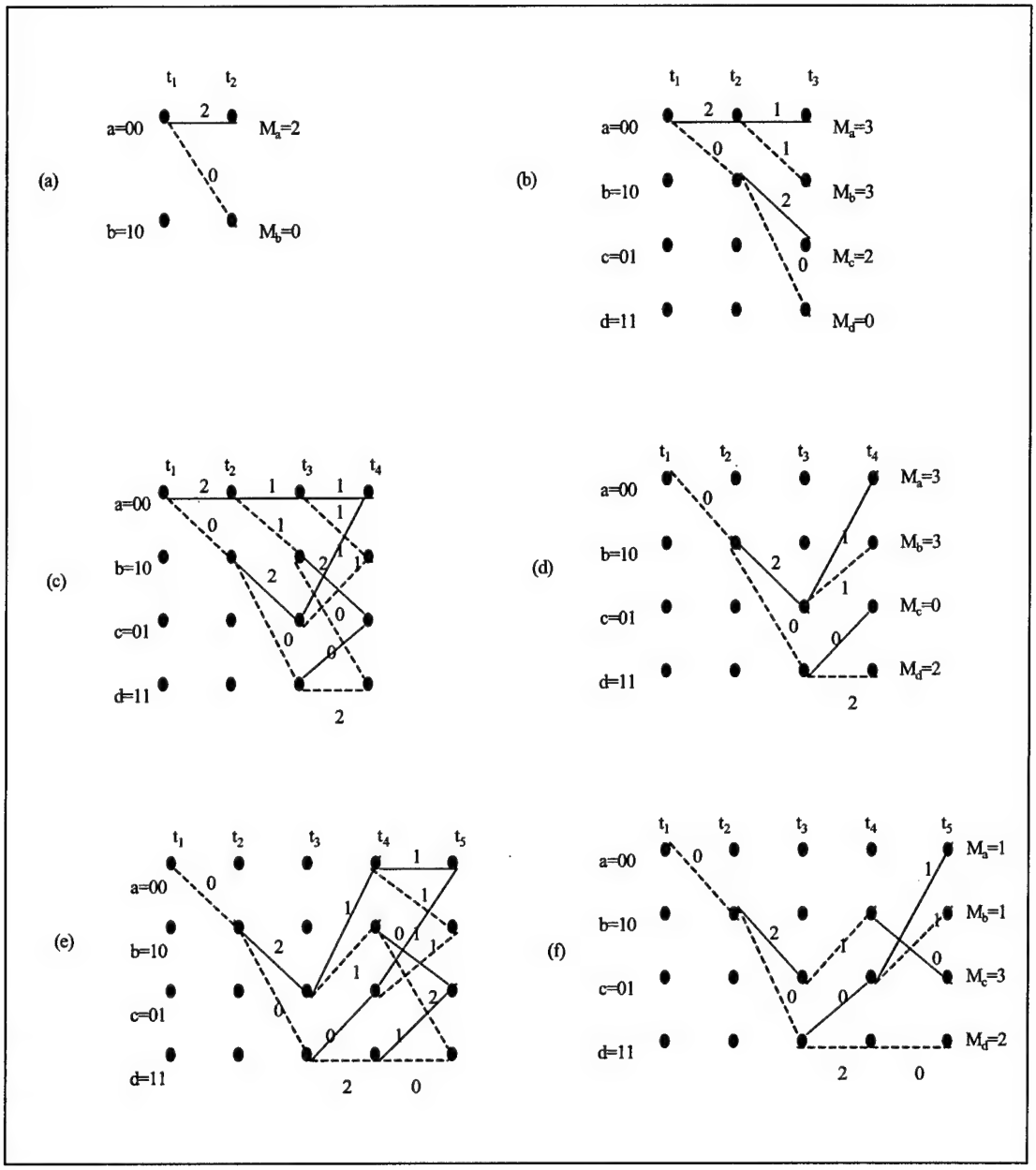

Figure 3-8 Selection of survivor paths. (a) Survivors at t_2. (b) Survivors at t_3. (c) Metric comparisons at t_4. (d) Survivors at t_4. (e) Metric comparisons at t_5. (f) Survivors at t_5 [11].

3.4.2 *Viterbi Algorithm with Soft Decision Outputs*

The Viterbi algorithm (VA) has become a standard tool in communication receivers. Recently, the number of applications that use two VA's in a concatenated way has been increasing. When using these concatenation schemes (serial or parallel), there are two draw backs: first, the VA produces burst of errors; and second the VA produces hard decisions prohibiting the other concatenated decoder from using its capability to accept soft decisions. The first drawback can be eliminated by using some interleaving between the two concatenated decoders. To eliminate the second drawback, the first VA decoder needs to output soft decisions; *i.e.*, reliability information. This should improve the performance of the other decoder.

The VA is modified in [51] to deliver not only the most likely path sequence, but also either the *a posteriori* probability for each bit or a reliability value. This modified algorithm of VA is known as the Soft Output Viterbi Algorithm (SOVA).

The VA algorithm is usually derived as a maximum likelihood sequence estimator. By re-deriving a more general VA that includes the *a priori* information to maximize the *a posteriori* probability, the metrics used in the previous section to incorporate the *a priori* or *a posteriori* information about the probabilities of the information bits can be modified accordingly. The modified metric of the VA can be derived in its MAP form. Recall from Equation 3-26 that the algorithm search for the state sequence, m', satisfies the following equation up to time j:

$$S^{(m')} = \arg\left\{\max_{m} \; M_j^{(m)}\right\} \tag{3-54}$$

where $M_j^{(m)}$ represent the accumulated branch metric up to time j and it is given by:

$$M_j^{(m)} = P(Y|S^{(m)})\ P(S^{(m)}) \tag{3-55}$$

We can write $P(Y|S^{(m)})$ in terms of its individual components up to time j:

$$P(Y|S^{(m)}) = \prod_{i=0}^{j} P(y_i|s_{i-1}^{(m)}, s_i^{(m)})$$

$$= P(y_j|s_{j-1}^{(m)}, s_j^{(m)}) \prod_{i=0}^{j-1} P(y_i|s_{i-1}^{(m)}, s_i^{(m)})$$

$$= P(y_j|x_j^{(m)}) \prod_{i=0}^{j-1} P(y_i|x_i^{(m)}) \tag{3-56}$$

Similarly, $P(S^{(m)})$ can be written as:

$$P(S^{(m)}) = \prod_{i=0}^{j} P(s_i^{(m)}|s_{i-1}^{(m)})$$

$$= P(s_j^{(m)}|s_{j-1}^{(m)}) \prod_{i=0}^{j-1} P(s_i^{(m)}|s_{i-1}^{(m)})$$

$$= P(m_j^{(m)}) \prod_{i=0}^{j-1} P(m_i^{(m)}) \tag{3-57}$$

Substituting Equations 3-56 and 3-57 into Equation 3-54 yields:

$$S^{(m')} = \arg\left\{\max_m \prod_{i=0}^{j-1} P(y_i|x_i^{(m)}) \prod_{i=0}^{j-1} P(m_i^{(m)})\ P(m_j^{(m)})\ P(y_j|x_j^{(m)})\right\} \tag{3-58}$$

In Equation 3-58, the maximum is not changed if:

1. Take the logarithm;

2. Multiply by 2;

3. Add two constants which are independent on the message, m.

By taking the logarithm of the right hand side of Equation 3-58 this yields:

$$M_j^{(m)} = \sum_{i=0}^{j-1} \log P(y_i | x_i^{(m)}) + \sum_{i=0}^{j-1} \log P(m_i^{(m)}) + \log P(m_j^{(m)}) + \log P(y_j | x_j^{(m)}) \qquad (3\text{-}59)$$

In the case of a $1/N$ coded scheme, Equation 3-59 can be written as:

$$M_j^{(m)} = \sum_{i=0}^{j-1} \sum_{n=0}^{N-1} \log P(y_{i,n} | x_{i,n}^{(m)}) + \sum_{i=0}^{j-1} \log P(m_i^{(m)}) + \log P(m_j^{(m)}) + \sum_{n=0}^{N-1} \log P(y_{j,n} | x_{j,n}^{(m)}) \qquad (3\text{-}60)$$

The accumulated branch metric up to time (j-1) can be defined by $M_{j-1}^{(m)}/2$ and it is given by:

$$M_{j-1}^{(m)}/2 = \sum_{i=0}^{j-1} \sum_{n=0}^{N-1} \log P(y_{i,n} | x_{i,n}^{(m)}) + \sum_{i=0}^{j-1} \log P(m_i^{(m)}) \qquad (3\text{-}61)$$

Substituting Equation 3-61 into Equation 3-60, multiplying by 2, and adding two constants $-C_m$ and $-C_y$, gives:

$$M_j^{(m)} = M_{j-1}^{(m)} + [2 \log P(m_j^{(m)}) - C_m] + [\sum 2 \log P(y_{j,n} | x_{j,n}^{(m)}) - C_y] \qquad (3\text{-}62)$$

where C_m and C_y are independent of message m and are defined by:

$$C_m = \log P(m_j = +1) + \log P(m_j = -1) \qquad (3\text{-}63)$$

$$C_y = \log P(y_{j,n} | x_{j,n} = +1) + \log P(y_{j,n} | x_{j,n} = -1) \tag{3-64}$$

Further simplification of the individual components of Equation 3-62 produces:

$$2\log P(m_j^{(m)}) - C_m = 2\log P(m_j^{(m)}) - \log P(m_j = +1) - \log P(m_j = -1) \tag{3-65}$$

Note that in the analysis, $m_j^{(m)}$ can take on +1or -1 values:

- In the case of $m_j^{(m)} = +1$, Equation 3-65 simplifies to:

$$2\log P(m_j^{(m)}) - C_m = 2\log P(m_j = +1) - \log P(m_j = +1) - \log P(m_j = -1)$$

$$= \log P(m_j = +1) - \log P(m_j = -1)$$

$$= \log \frac{P(m_j = +1)}{P(m_j = -1)} \tag{3-66}$$

- In the case of $m_j^{(m)} = -1$, Equation 3-65 simplifies to:

$$2\log P(m_j^{(m)}) - C_m = 2\log P(m_j = -1) - \log P(m_j = +1) - \log P(m_j = -1)$$

$$= \log P(m_j = -1) - \log P(m_j = +1)$$

$$= -\log \frac{P(m_j = +1)}{P(m_j = -1)} \tag{3-67}$$

From Equations 3-66 and 3-67, Equation 3-65 is generalized by:

$$2\log P(m_j^{(m)}) - C_m = m_j^{(m)} \log \frac{P(m_j = +1)}{P(m_j = -1)} \tag{3-68}$$

Similarly,

$$\sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n}^{(m)}) - C_y$$

$$= \sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n}^{(m)}) - \log P(y_{j,n}|x_{j,n} = +1) - \log P(y_{j,n}|x_{j,n} = -1) \qquad (3\text{-}69)$$

Note that $x_{j,n}^{(m)}$ can take on +1or -1 values:

- In the case of $x_j^{(m)} = +1$, Equation 3-69 becomes:

$$\sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n}^{(m)}) - C_y$$

$$= \sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n} = +1) - \log P(y_{j,n}|x_{j,n} = +1) - \log P(y_{j,n}|x_{j,n} = -1)$$

$$= \sum_{n=0}^{N-1} \log P(y_{j,n}|x_{j,n} = +1) - \log P(y_{j,n}|x_{j,n} = -1)$$

$$= \sum_{n=0}^{N-1} \log \frac{P(y_{j,n}|x_{j,n} = +1)}{P(y_{j,n}|x_{j,n} = -1)} \qquad (3\text{-}70)$$

- In the case of $x_j^{(m)} = -1$, Equation 3-69 becomes:

$$\sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n}^{(m)}) - C_y$$

$$= \sum_{n=0}^{N-1} 2\log P(y_{j,n}|x_{j,n} = -1) - \log P(y_{j,n}|x_{j,n} = +1) - \log P(y_{j,n}|x_{j,n} = -1)$$

$$= \sum_{n=0}^{N-1} \log P(y_{j,n} | x_{j,n} = -1) - \log P(y_{j,n} | x_{j,n} = +1)$$

$$= -\sum_{n=0}^{N-1} \log \frac{P(y_{j,n} | x_{j,n} = +1)}{P(y_{j,n} | x_{j,n} = -1)} \tag{3-71}$$

From Equations 3-70 and 3-71, a general form of Equation 3-69 is derived:

$$\sum_{n=0}^{N-1} 2 \log P(y_{j,n} | x_{j,n}^{(m)}) - C_y = \sum_{n=0}^{N-1} x_{j,n}^{(m)} \log \frac{P(y_{j,n} | x_{j,n} = +1)}{P(y_{j,n} | x_{j,n} = -1)} \tag{3-72}$$

Substituting Equations 3-68 and 3-72 into Equation 3-62 yields:

$$M_j^{(m)} = M_{j-1}^{(m)} + \sum_{n=0}^{N-1} x_{j,n}^{(m)} \log \frac{P(y_{j,n} | x_{j,n} = +1)}{P(y_{j,n} | x_{j,n} = -1)} + m_j^{(m)} \log \frac{P(m_j = +1)}{P(m_j = -1)} \tag{3-73}$$

For the Additive White Gaussian Noise channel with fading, Equation 3-73 is simplified further by:

$$p(y_{j,n} | x_{j,n} = +1, a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(y_{j,n} - a_{j,n})^2}{2\sigma^2}} \tag{3-74}$$

$$p(y_{j,n} | x_{j,n} = -1, a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(y_{j,n} + a_{j,n})^2}{2\sigma^2}} \tag{3-75}$$

with $\sigma^2 = \frac{N_0}{2E_s}$,

$$\log\frac{P(y_{j,n}|x_{j,n}=+1)}{P(y_{j,n}|x_{j,n}=-1)}=4\frac{E_s}{N_0}a_{j,n}y_{j,n}$$

$$=L_c y_{j,n} \tag{3-76}$$

where $L_{c_{j,n}}$ is known as the reliability of the channel and it is given by:

$$L_{c_{j,n}}=4\frac{E_s}{N_0}a_{j,n} \tag{3-77}$$

Also, the log-likelihood of the *a priori* information m_j is defined by $L(m_j)$ and it is given by:

$$L(m_j)=\log\frac{P(m_j=+1)}{P(m_j=-1)} \tag{3-78}$$

Substituting Equations 3-76 and 3-78 into Equation 3-73, allow for the simplification of the accumulated branch metric using the soft values of the information channel as [87, 88]:

$$M_j^{(m)}=M_{j-1}^{(m)}+\sum_{n=0}^{N-1}x_{j,n}^{(m)}\ L_{c_{j,n}}y_{j,n}+m_j^{(m)}\ L(m_j) \tag{3-79}$$

The final accumulated metric given in Equation 3-79 represents the path metric up to time *j*. From here, the regular process of the VA terminates and the process of reliability calculation in the SOVA version begins. At time *j*, the probability of path *m* is proportional to the exponential of the accumulated branch metric. The relation is given by:

$$P(path\ m)\ \alpha\ e^{M_j^{(m)}/2} \tag{3-80}$$

The VA algorithm proceeds in the usual way by calculating the metrics for the path m using Equation 3-79 with or without $L(m)$. For each state, it selects the path with the largest metric $M_j^{(m)}$. Figure 3-9 shows a trellis example where the VA has selected the maximum likelihood sequence, m', and has discarded the sequence m at time j. In Figure 3-9, $\Delta^l{}_j$ denotes the metric difference at time j in l nonsurvivor paths, and $\delta + 1$ is the number of nonsurvivor paths.

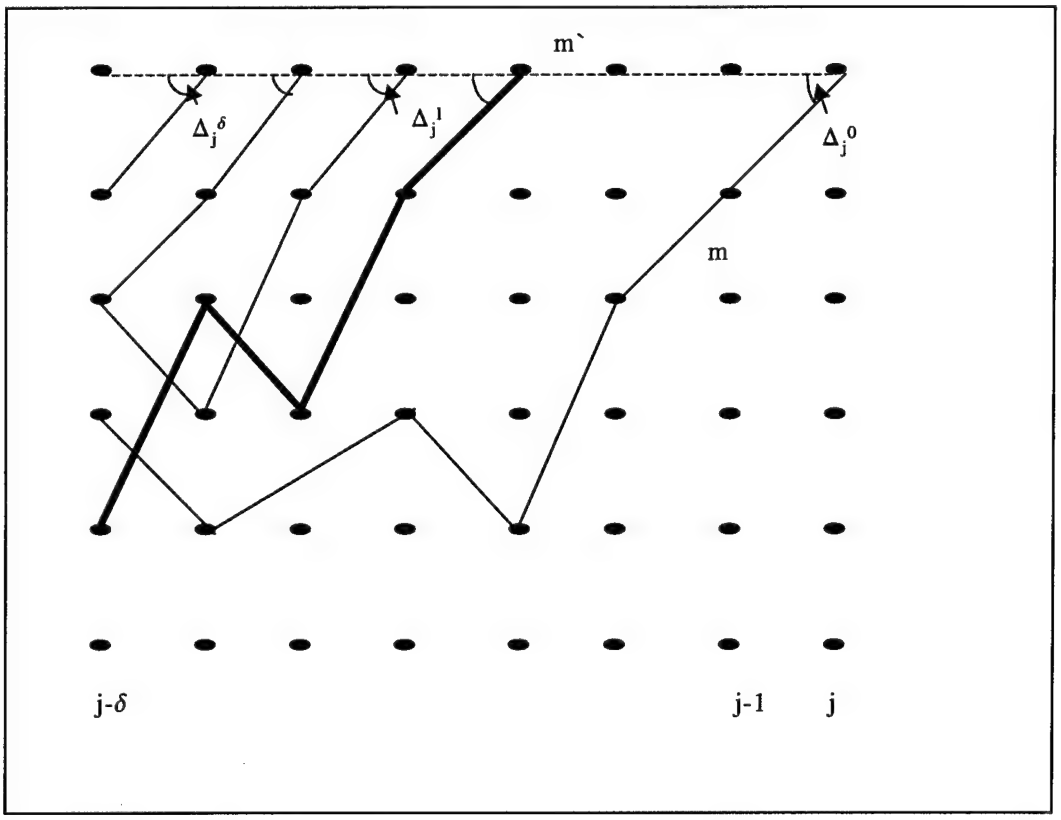

Figure 3-9 Example with two metric differences for the derivation of the traceback SOVA [51].

The probability of choosing m' is the probability of choosing the correct sequence. It also holds that $P(correct)+P(incorrect)=1$. Normalizing $P(path\ m)$ and $P(path\ m')$ yields:

$$P(correct)=\frac{P(path\ m')}{P(path\ m)+P(path\ m')}$$

$$=\frac{e^{M_j^{(m')}/2}}{e^{M_j^{(m')}/2}+e^{M_j^{(m)}/2}}$$

$$=\frac{e^{\Delta_j}}{1+e^{\Delta_j}} \tag{3-81}$$

where Δ_j denotes the metric difference and it is given by:

$$\Delta_j=\frac{1}{2}\left(M_j^{(m')}-M_j^{(m)}\right) \tag{3-82}$$

The likelihood ratio or soft value of this binary path decision is:

$$\log\frac{P(correct)}{1-P(correct)}=\log\left[\frac{\left(e^{\Delta j}+1\right)\ e^{\Delta j}}{1+e^{\Delta_j}}\right] \tag{3-83}$$

By taking the logarithm in Equation 3-83 for the base *e*, then:

$$\log\frac{P(correct)}{1-P(correct)}=\ln\left[e^{\Delta j}\right]$$

$$=\Delta_j \tag{3-84}$$

Then, along the maximum likelihood path m', which decides the bit $\hat{m}_{j-\delta}$, $\delta+1$ non-surviving paths with indices $l=0,...,\delta$ have been discarded. The difference between their metrics is $\Delta_j^l \geq 0$. If the bit $\hat{m}_{j-\delta}^l$ on the discarded path equals the decided bit $\hat{m}_{j-\delta}$, then no bit error is made if the discarded path was selected. Thus, the reliability of this bit is infinite. Otherwise, if the bits differ, i.e., $e_{j-\delta}^l = m_{j-\delta}^l \oplus \hat{m}_{j-\delta} = -1$, the log-likelihood value of the bit error $e_{j-\delta}^l$ equals Δ_j^l. Consequently, we have:

$$L(e_{j-\delta}^l) = \log \frac{P(e_{j-\delta}^l = +1)}{P(e_{j-\delta}^l = -1)} = \begin{cases} \infty & m_{j-\delta}^l = \hat{m}_{j-\delta} \\ \Delta_j^l & m_{j-\delta}^l \neq \hat{m}_{j-\delta} \end{cases} \tag{3-85}$$

The total error resulting from all possibly discarded paths for bit $\hat{m}_{j-\delta}$ is given by:

$$e_{j-\delta} = \sum_{l=0}^{\delta} \oplus e_{j-\delta}^l \tag{3-86}$$

If the Δ_j^l and the e_j^l are statistically independent with respect to the indices, then the log-likelihood ratio of the decisions (the soft output of the VA (SOVA)), is the decision $\hat{m}_{j-\delta}$ times the L-values of the errors. This is shown by:

$$L(\hat{m}_{j-\delta}) = \hat{m}_{j-\delta} \sum_{l=0}^{\delta} [+]\ L(e_{j-\delta}^l)$$

$$\approx \hat{m}_{j-\delta} \min_{l=0,...,\delta} \Delta_j^l \tag{3-87}$$

The minimum in Equation 3-87 has to be taken only over those indices l where the bits differ. Therefore, we have the same decisions as with the classical VA. The

reliability of these decisions is obtained by taking the minimum of the metric differences along the maximum likelihood path whenever the update sequences $e_{j-\delta}^{l}$ indicate this. As a result, the modified VA accepts log-likelihood (soft) L-values from both the a priori information and from the channel and delivers such values for the output bits.

3.5 Summary

In this chapter, the principle of the iterative decoding process of the turbo code decoder was presented. The turbo code decoder mainly depends on the soft-input/soft-output constitute decoders. The chapter also described how to apply the soft-in/soft-out constituent decoder to the iterative turbo decoder. The information transfer from one decoder to another is essential for improving the performance from one step to the next step in the iterative decoding process. The Viterbi algorithm was also reviewed, along with its modified version that delivers soft decisions used by a constituent soft-input/soft-output decode in the iterative decoding. All related mathematics associated with implementing the iterative decoding process were presented. This was done to provide a better understanding of the process and how information is passed through iterations.

4 Methodology

4.1 Introduction

The research methodologies applied throughout this dissertation are mathematical analysis and computer simulations. This chapter presents the methodology used to develop the mathematical models of the analytical bound for turbo code performance. This chapter also covers the development for the simulation model of the proposed turbo code. The algorithm use to build a simulation model of the channel under consideration is also discussed.

The software packages MathCAD® and MATLAB® are used to accomplish complicated calculations. The simulation models are implemented using the software package MATLAB®. The results of the mathematical analysis and computer simulation are displayed using graphs in which the bit error rate is plotted versus signal-to-noise ratio (E_b/N_0) in decibels. Tables are also used to display results.

Section 4.2 presents the development of the analytical model to evaluate the performance of turbo codes using the union bound. Application of the uniform interleaver to the union bound to facilitate its calculations is presented. A discussion of the calculation of the average weight enumerating function of the parallel concatenation

is also presented. Section 4.3 presents the development of the simulation model used to implement the turbo encoding, the channel, and the turbo code decoding. The models that describe the behavior of the Rayleigh fading channel are also presented. The algorithm to implement the soft out Viterbi algorithm (SOVA) and the practical consideration is discussed. Section 4.4 summarizes the contents of this chapter.

4.2 Analytic Model

Turbo code performance in high signal-to-noise ratio regions is best evaluated using the analytical bound. This is due to the long execution times of computer simulations. It is useful to consider turbo codes as block codes. The input sequences are restricted to length N, where N corresponds the interleaver size in the turbo code encoder. The union upper bound is a popular method of bounding block code performance given the code weight distribution. For turbo codes, deriving the weight distribution for a particular interleaving scheme is very difficult. The authors in [61, 62] introduced the idea of averaging the weight enumerating function using a uniform interleaver. In this section, the average bound calculation details, as applied to turbo codes, is presented. For discussion, it is assumed that the component codes are fixed convolutional codes with certain memory.

4.2.1 Union Bound

For a $(N, 3N)$ turbo code with an interleaver of length N bits, without loss of generality, it is assumed that the all-zero codeword is sent. For discussion, consider a traditional union upper bound with maximum likelihood decoding of the probability of bit error [63]:

$$P_{bit} \leq \sum_{j} \sum_{w} \frac{w}{N} A_{w,j}^{c} P_2(j) \tag{4-1}$$

where $A_{w,j}^{c}$ denotes the number of codewords of the parallel concatenation (turbo code) with weight j generated by a word of information weight w. Parameters d_1 and d_2 represent the parity weights of the first and second encoders respectively. The weight $j = d_1 + d_2 + w$ and $P_2(j)$ is the pair-wise error probability of error between the zero-codeword and the codeword with weight j. The conditional weight enumerating function, $A_w^c(Z)$, enumerates all the codewords of weight j generated by a word with information weight w. It is given by:

$$A_w^c(Z) = \sum_{j} A_{w,j}^{c} \; Z^j \tag{4-2}$$

where Z is dummy variable. For turbo codes with a fixed interleaver, the construction of $A_w^c(Z)$ requires an exhaustive search. Due to the complexity involved in this search, the authors in [2] introduced the concept of the uniform interleaving to reduce search overhead. Using this approach, an average upper bound on performance can be constructed by averaging over all possible interleavers.

An N length uniform interleaver is a probabilistic device representing the average behavior of all deterministic bit interleavers. For each input sequence having information weight w and length N, the uniform interleaver maps the input sequence to all its distinct sequences, N_{cw}, of weight w and length N with equal probability $1/N_{cw}$.

From this, the average conditional WEF of the parallel concatenation (turbo code) can be obtained from the WEF of the constituent codes, $A_w^{C_1}(Z)$ and $A_w^{C_2}(Z)$:

$$A_w^{C_p}(Z) = \frac{A_w^{C_1}(Z)\ A_w^{C_2}(Z)}{N_{cw}}, \qquad N_{cw} = \binom{N}{w} = \frac{N!}{(N-w)!w!} \tag{4-3}$$

Example: For an input frame m = 0101 of length N = 4 and weight w = 2, all distinct possible mappings of this input frame are given in Figure 4-1. As shown in Figure 4-1, there are N_{cw} = 6 possible interleaving maps. The uniform interleaver would map this input sequence to all of these distinct sequences with probability $P = 1/N_{cw} = 1/6$.

Using the uniform interleaver, each input word of weigh w at the input of the first encoder $C1$ is mapped into all its distinct permutations, which, through the second encoder $C2$, generates all codewords corresponding to the input weight w. As a consequence, all input words of the same weight generate the same set of codewords. From this, the conditional weight enumerating functions of $C1$ and $C2$ become independent and can be multiplied to yield the overall conditional weight enumerating function of the parallel concatenated code using Equation 4-3.

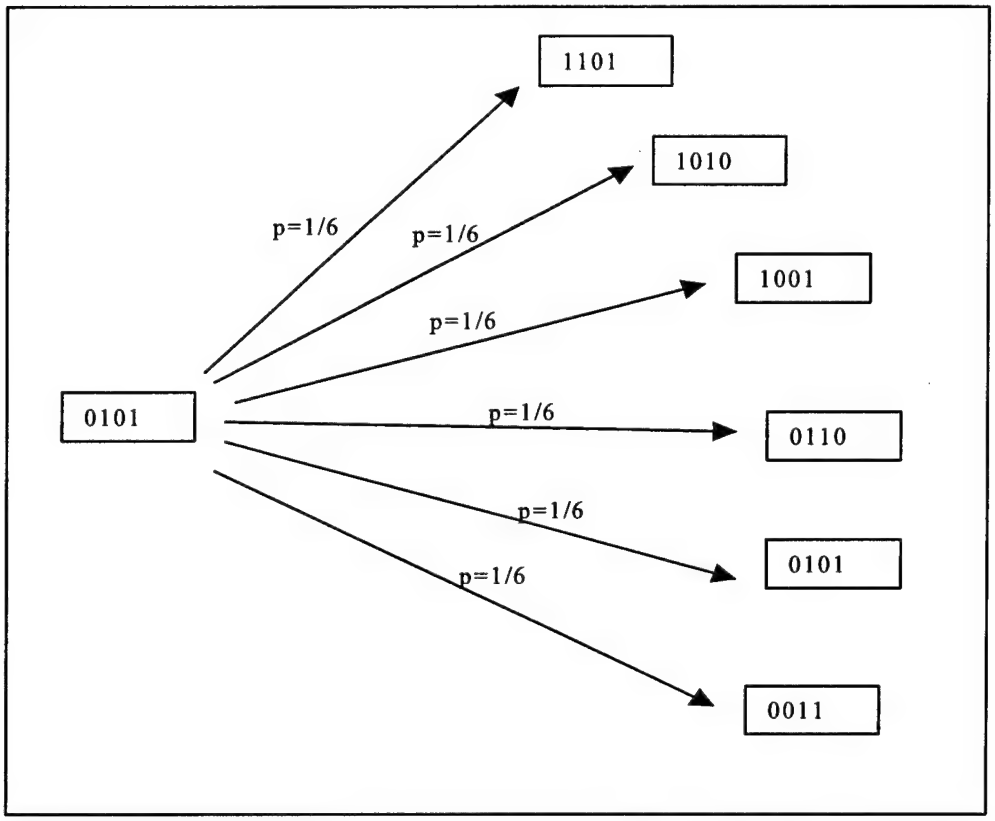

Figure 4-1 All possible interleaved versions of the input frame, $m = 0101$.

4.2.2 Computation of the Conditional Weight Enumerating Function

Like the probability of bit error calculations for convolutional codes using the transfer function bound technique, it is assumed here the all-zero codeword is transmitted and the probability the decoder selects some alternative codeword is calculated. Calculating this bound assumes that the weight enumerating function (WEF) of the turbo code is known. The transfer function bound and the WEF for turbo codes differ from those of convolutional codes in four aspects [62]:

1. Turbo code bounds (and then WEF) require a term-by-term joint enumerator for all possible combinations of input weight and output weights of both encoders.

2. Because turbo codes are block codes, it is necessary to accurately enumerate *compound* error events that can include more than one excursion from the all-zero state during the fixed block length.

3. Since using one specific interleaver is intractable, the bound is developed as a random coding bound using the uniform interleaver.

4. Turbo code bound calculations assume maximum likelihood decoding, whereas the iterative decoding procedure used in practice is not guaranteed to converge to the maximum likelihood decoder.

From the knowledge of the individual terms of each convolutional weight enumerating function for the two convolutional codes, $A_w^{C_1}(Z)$ and $A_w^{C_2}(Z)$, the WEF of the parallel concatenation can be enumerated jointly term-by-term for all possible combinations of input and output weights using Equation 4-3. Using this information, a recursive algorithm to compute the weight enumerating function of a convolutional code can be explained in detail.

The operation of any convolutional code can be completely described by its state diagram. For illustration, a convolutional code with octal notation $(5/7)_8$ is used. This convolutional encoder is shown in Figure 4-2 and its state diagram is shown in Figure 4-3.

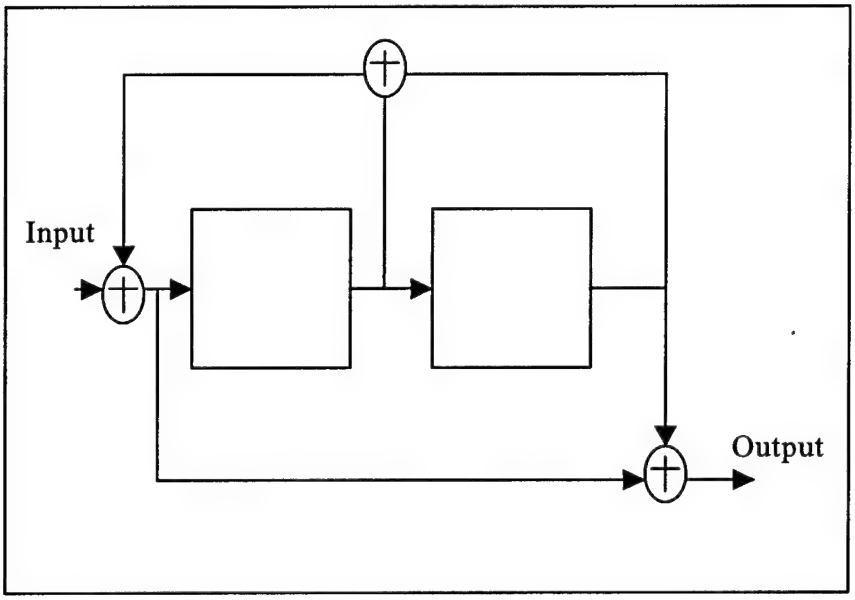

Figure 4-2 Convolutional encoder with $(5/7)_8$.

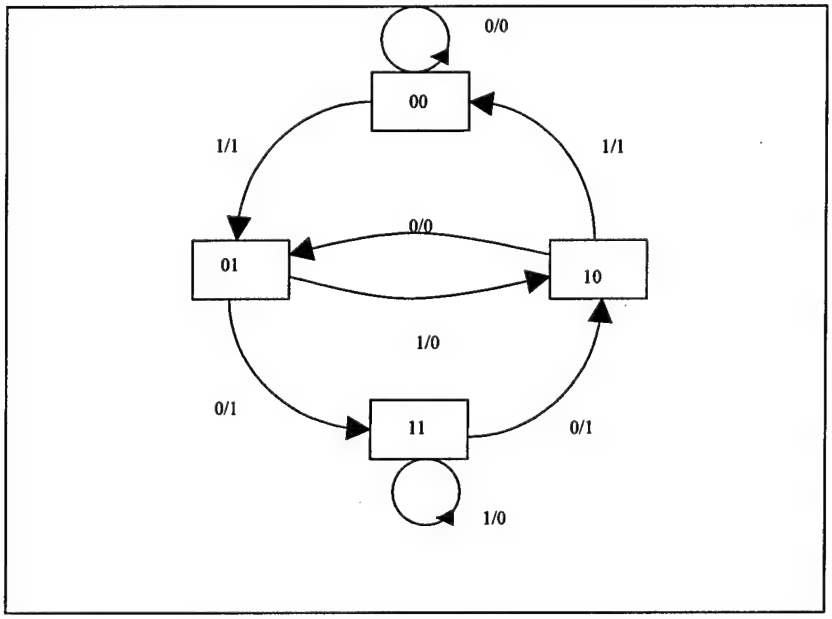

Figure 4-3 State diagram of $(5/7)_8$ convolutional encoder.

In Figure 4-3, each transition between states is labeled by the input information bit and the corresponding output encoded bit. It is convenient to replace each edge label in Figure 4-3 with a monomial $L^l I^i D^d$, where l is always equal to 1 (represents the accumulated length), and i and d are either 0 or 1 (each represent the input and output weights, respectively), depending on whether the corresponding input and output are 0 or 1, respectively. Then, the information in the state diagram can be summarized by the state transition matrix, $S(L, I, D)$, as given below:

$$S(L,I,D)=\begin{pmatrix} L & LID & 0 & 0 \\ 0 & 0 & LI & LD \\ LID & L & 0 & 0 \\ 0 & 0 & LD & LI \end{pmatrix} \tag{4-4}$$

where each entry $L^l I^i D^d$ shows an l step transition from the state "row number" to the state "column number" with an input weight of i that results in an output weight of d. A "0" as the entry means the transition is not possible. In writing the one-step transition matrix above, the states are arranged as the state "00" is the first state, the state "01" is the second state, the state "10" is the third state, and the state "11" is the last state. If we denote the transfer function of the code by $T(L, I, D)$, where $T(L, I, D)$, is given by:

$$T(L,I,D)=\sum_{l\geq 0}\sum_{i\geq 0}\sum_{d\geq 0} L^l I^i D^d \; t(l,i,d) \tag{4-5}$$

where $t(l, i, d)$ denotes the number of paths (codewords) of length l, input weight i, and output weight d, starting and ending in the zero state. Using the method described in [89] and the approximation made in [62] about ignoring the termination term, the transfer

function of the of the code $T(L, I, D)$ is the first row and first column entry of the inverse of this matrix $[\boldsymbol{I} - S(L, I, D)]$:

$$T(L, I, D)=\left[\ \boldsymbol{I} - S(L, I, D)\ \right]_{00,00} \tag{4-6}$$

where $\boldsymbol{I}$ denotes the identity matrix. Using the MathCAD® software package, Equation 4-6, can be calculated. From [62], the first row first column element represents the transfer function of the code and is given by:

$$T(L,I,D)\approx\frac{1-LI-L^2I-L^3(D^2-I^2)}{1-L(1+I)-L^3(D^2-I-I^2+I^3D^2)+L^4(D^2-I^2-I^2D^4+I^4D^2)} \tag{4-7}$$

Multiplying both sides of Equation 4-7 by the denominator of the right-hand side and substituting into Equation 4-5 yields:

$$\sum_l\sum_i\sum_d L^lI^iD^d\ t(l,i,d)\left[1-L(1+I)-L^3(D^2-I-I^2+I^3D^2)+L^4(D^2-I^2-I^2D^4+I^4D^2)\right]$$

$$=1-LD-L^2D+L^3(D^2-I^2) \tag{4-8}$$

Equating the coefficients of $t(l, i, d)$ of both sides of Equation 4-8,. the following recursion is formed for $t(l, i, d)$, for $l\geq 0$, $i\geq 0$, $d\geq 0$:

$$\begin{aligned}t(l,i,d)=&\,t(l-1,i-1,d)+t(l-1,i,d)\\&+t(l-3,i-3,d-2)-t(l-3,i-2,d)-t(l-3,i-1,d)+t(l-3,i,d-2)\\&-t(l-4,i-4,d-2)+t(l-4,i-2,d-4)+t(l-4,i-2,d)-t(l-4,i,d-2)\\&+\delta(l,i,d)-\delta(l-1,i-1,d)-\delta(l-2,i-1,d)-\delta(l-3,i,d-2)+\delta(l-3,i-2,d)\end{aligned} \tag{4-9}$$

where $\delta(l,i,d)=1$ if $l=i=d=0$ and $\delta(l,i,d)=0$ otherwise. The initial conditions hold that $t(l,i,d)=0$ if any index is negative. Note that all the terms in Equation 4-9 assume that the starting and ending state is the zero state. By taking the first row, first column entry of the matrix $[\boldsymbol{I} - S]^{-1}$ as the proper choice, we terminate both encoders in the simulation model.

Using the recursion formula in Equation 4-9, the terms of the weight enumerating function, $A_{w,d}^{C}$, of the convolutional code C with codeword of length N is given by the following:

$$A_{w,d}^{C} = t(l = N, I = w, D = d) \tag{4-10}$$

From Equations 4-2, 4-3, and 4-10, the average weight enumerating function of the parallel concatenation can be calculated.

4.2.3 Analytical Bound Calculations and Validation

In computing the union bound given in Equation 4-1, there is an observation about the convergence of the bound at low signal-to-noise ratios (E_b/N_0 less than R_0, where R_0 is the computational cut off rate). The bound suddenly diverges to a useless bound greater than 1. On the other hand, when E_b/N_0 is above R_0, the false convergence is not a problem. Only a small number of terms in the summation are needed for convergence of the bound (i less than 10 is good) and this is almost independent of the value of N.

The method explained in the previous section is used to evaluate the upper bound for turbo codes. Figure 4-4 shows the bound P_{bit} for turbo code $(1, 5/7, 5/7)$ in the Additive White Gaussian Noise (AWGN) channel for various block length N. Note that the transition from a well-behaved, useful (below 2.03 dB), low-P_{bit} bound into a diverged, useless bound greater than 1 occurs if the block length N becomes larger (the same as observed in [62], which validates our model). The abrupt transition occurs roughly when the information bit signal-to-noise ratio (E_b/N_0) drops below the threshold determined by the computational cutoff rate R_0 (*i.e.*, when $E_s/N_0 = r\,E_b/N_0 < -\ln\left(2^{1-r} - 1\right)$ for a code rate r [90]).

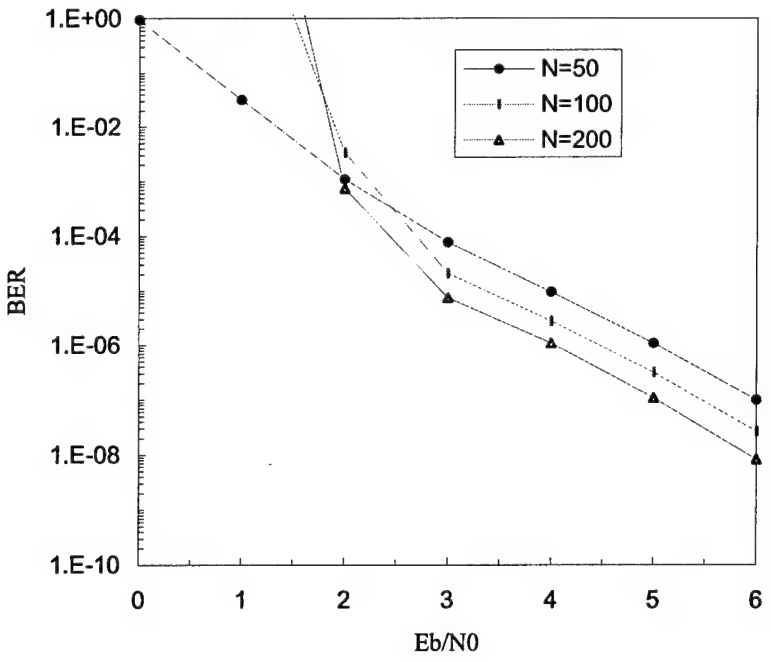

Figure 4-4 Bounds on P_{bit} for various block lengths N for the $(1, 5/7, 5/7)$ turbo code.

To validate the analytic model with the simulation model, simulation results have been compared with the computed bound in Figure 4-5 for a frame length of 256 bits. The simulated model is based on two identical RSC with generators $(5/7)_8$ and rate 1/3. The SOVA algorithm in [51] is used as a component decoder with 8 iterations. From the graph, as also observed in [62], it is observed that above the R_0 threshold of 2.03 dB, the simulated turbo decoder bit-error rate closely matches the error predicted by the bound. Below this threshold, the turbo decoder experiences its own region of "divergence" wherein its performance deteriorates rapidly because its iterative decoding algorithm frequently fails to converge. However, this "divergence" is far less steep than that experienced by the bound. It occurs well below the R_0 threshold, allowing turbo decoders to operate in the region between the limit determined by channel capacity and the limit determined by R_0.

Unfortunately, the region where the turbo codes have offered astounding performance is below the computational cutoff rate threshold, so at first glance the bounds appear to be of dubious utility. For E_b/N_0 above the computational cutoff rate threshold, the bound is not only meaningful, but essentially tells the whole story (i.e., the bit-error rate predicted by the bound is accurately achieved both by a maximum-likelihood decoder and by a turbo decoder). This is demonstrated by the confluence of the simulation and bound performance curves in Figure 4-5 at high E_b/N_0. In this region, evaluation of the bound requires only a few terms in the summation. The bound's behavior is predictable from the heuristics analysis about the relationships of weight distribution, permutation, and the number of codes reported in [41].

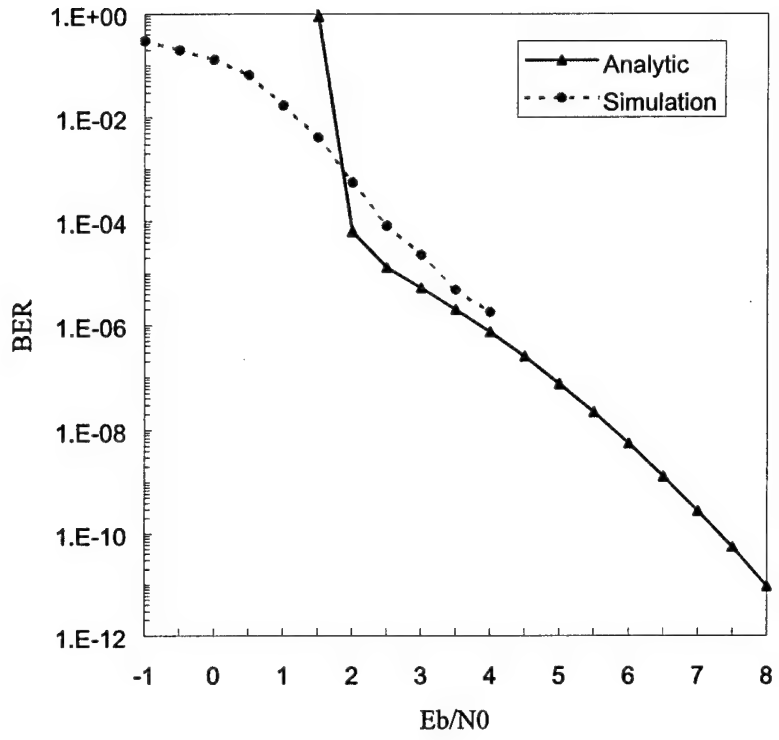

Figure 4-5 Analytical error bounds versus simulated bit error rates for frame length of 256 bits.

4.3 Simulation Model

The simulation of communication systems requires a representation of the system. The standard description of the system is the block diagram, where each block represents a signal-processing operation. The block diagram of the simulation model used in this investigation is shown in Figure 4-6.

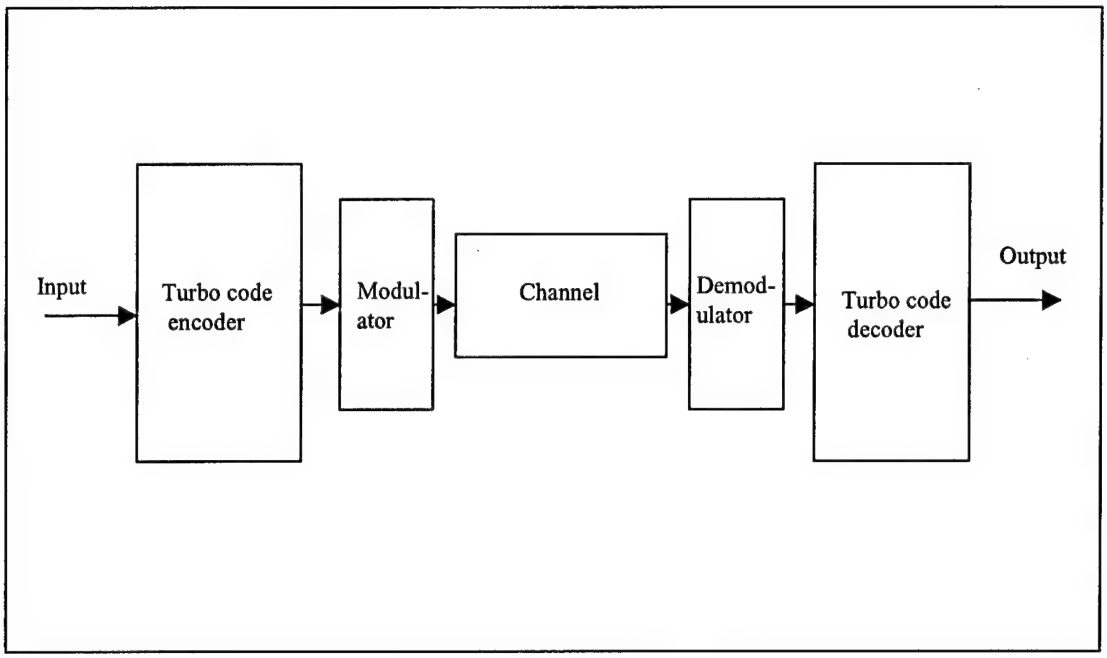

Figure 4-6 Block diagram of the simulation model.

Each block shown in Figure 4-6 contains the algorithms and equations needed to implement the block functions within the simulation. These individuals blocks represent subsystems of the overall system.

Using a hierarchical modeling approach, a simulation-based analysis of communication systems is possible. This hierarchical modeling allows a complex system to be modeled. In a hierarchical modeling environment, complex models can be built up from very simple building blocks. This modularity allows for ease of code testing and evaluation. Figure 4-7 shows the implementation flow of the system model.

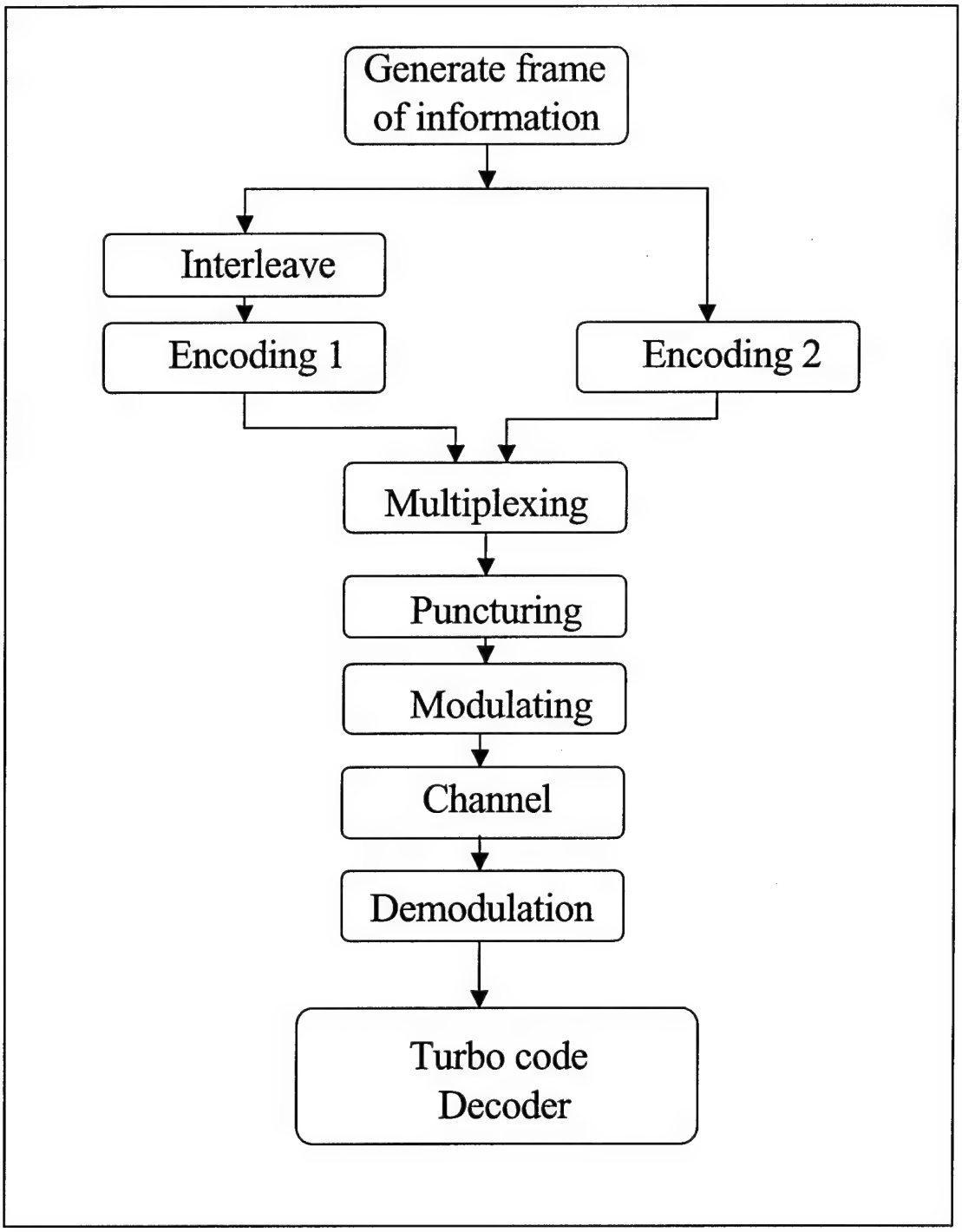

Figure 4-7 Steps of implementation of the simulation model.

As shown in Figure 4-7, the generation of information bits begins the process. The two encoders receive the same information bits, with the second encoder receiving the information bits after being permuted by an interleaver. The output of both encoders are multiplexed and punctured. Note that puncturing is an optional operation determined according to the required rate. The binary symbol outputs are used to modulate an RF carrier sinusoid. For this investigation, Binary Phase Shift Key (BPSK) modulation is used. Each code symbol results in the transmission of a pulse of carrier at either of two 180^0 separation phases. At the receiving end, the transmitted signal is faded and added to white Gaussian noise. It assumed that perfect synchronization occurs. The output of the front-end receiver (matched filter) at time i is given by:

$$y_i = a_i x_i \sqrt{E_s} + n_i \tag{4-11}$$

where y_i is the output of the matched filter, x_i is the transmitted symbol, x_i=±1 depending on whether the ith code symbol is 0 or 1, E_s is the energy per symbol, and n_i is a zero-mean Guassian noise with variance, $\sigma^2 = \sqrt{N_0/2}$. When coding is used, hard quantization of the received data usually entails a loss of about 2 db in E_b/N_0, where E_b is the energy per information bit, compared with infinitely fine quantization [91, 92]. Much of this loss can be recouped by quantizing y_i to 4 or 8 levels instead of merely 2. In the simulation model, the availability of high precision floating point arithmetic is assumed. In a real-time decoder, it is likely that the decoding algorithms would be implemented using fixed point arithmetic. The influence of quantization and fixed-point arithmetic for the implementation in turbo code decoder is presented in [93]. In [93], the authors

compared different decoder types and they concluded that the SOVA is the most computationally stable.

Now, the output of the matched filter is used as input to the turbo decoder. The turbo code decoder is implemented using two constituent decoders. Each decoder uses the SOVA algorithm. The entire procedure, from the generation of information bits until the decoder makes the decision, is repeated frame-by-frame according to the number of frames needed to be simulated or according to a predetermined number of accumulated errors needed.

4.3.1 Channel Model

A statistical model for the received signal envelope fading encountered on the wireless communication channel is useful for predicting the communication system performance. Signal fading can result from multipath, scattering, or diffraction. In general, fading is a time varying stochastic process. Its effect can be extended over a number of channel symbols. Coherence time, τ_c, is used to characterize a fade phenomenon and is intuitively regarded as the average time span that fading correlation lasts. In other words, any two bits that are separated by more than τ_c can be considered to be somewhat uncorrelated.

Under normal conditions, where all of the received multipath components of a symbol arrive within the symbol time duration, identical behavior in time is exhibited by the random fluctuations caused by fading to each spectral component of the signal. This

kind of fading is flat, or non-frequency selective, over the signal bandwidth. Therefore, it is called *flat fading*. In this thesis, flat fading is considered.

In wireless communication systems that adopt forward error correction (FEC) codes, this correlated property of a fading channel propagates a decoded bit error to subsequent bits, causing bursty errors and significantly degrading the performance. A channel interleaver is usually utilized to break the correlated channel disturbance into random channel symbol errors that are effectively corrected by FEC codes. As a result, the performance can be improved. The larger the interleaver size, the wider the correlated channel symbols can be separated, and consequently, the better performance.

In typical mobile-radio situations, buildings obstruct the direct line of sight between the transmitter and the receiver. In these cases, the mode of propagation of the electromagnetic energy from transmitter to receiver will be largely determined by way of scattering, either by reflection from flat sides of buildings or by diffractions around such buildings or other obstacles. Thus, the received signal results from the interference of many scattered radio paths between the base station and the mobile. Rayleigh fading characterizes these signal fluctuations. The Rayleigh fading model is of greatest interest and importance in the design of communication systems operating in fading environment.

Clarke [94] developed a random process model for the received power of random interfering waves. The model assumes a fixed transmitter with a vertically polarized antenna. The field incident on the mobile antenna is assumed to be composed of N equal amplitude azimuthal plane waves with arbitrary angles of arrival and arbitrary phases. Each incident wave undergoes a Doppler shift due to the motion of the receiver. Waves

arrive at the receiver at the same time, *i.e.,* no excess delay due to multipath is assumed (flat fading assumption). Due to the Doppler effect, the received frequency increases if the receiver is advancing into the approaching wave front and decreases if the receiver is moving away from the wave front. The maximum Doppler shift, f_d, is given by:

$$f_d = \frac{v.f_c}{C} \tag{4-12}$$

where v is the velocity of the mobile, f_c is the carrier frequency, and C is the velocity of electromagnetic radiation.

Let $T_c(t)$ and $T_s(t)$ be the summation of the received in-phase and quadrature components of the individual received waves, respectively. Thus, by the Central Limit Theorem, they are an uncorrelated zero-mean Guassian random process with equal variance given by:

$$E[T_c^2] = E[T_s^2] = E^2{}_0 / 2 \tag{4-13}$$

where E_0 is the real amplitude of the local average E-field. The envelope of the received E-field is:

$$a(t) = \sqrt{T_c^2(t) + T_s^2(t)} \tag{4-14}$$

Since T_c and T_s are Gaussian random variables, the received signal envelope a has a Rayleigh distribution. Assuming the angle of arrival to be uniformly distributed from 0 to 2π and an omnidirectional receiver, Clarke determined that the power spectral density $S(f)$ of the in-phase and quadrature components T_c and T_s is [14]:

$$S(f) = \frac{1.5}{\pi f_d \sqrt{1 - \left(\frac{f - f_c}{f_d}\right)^2}} \qquad |f - f_c| \le f_d \tag{4-15}$$

In the simulation model, the power spectrum in the baseband is implemented so the frequency considered is $freq = f - f_c$. The normalized temporal correlation of the received amplitudes, which correspond to this spectral density given in Equation 4-15 is given by:

$$R(t) = J_0(2\pi f_d t) \tag{4-16}$$

where $J_0(*)$ is the zeroth order Bessel function of the first kind and t is the time separation. The power spectral density given in Equation 4-15 is used to shape the output fading amplitudes of the simulated channel [14]. Implementation of this fading model is shown in Figure 4-8.

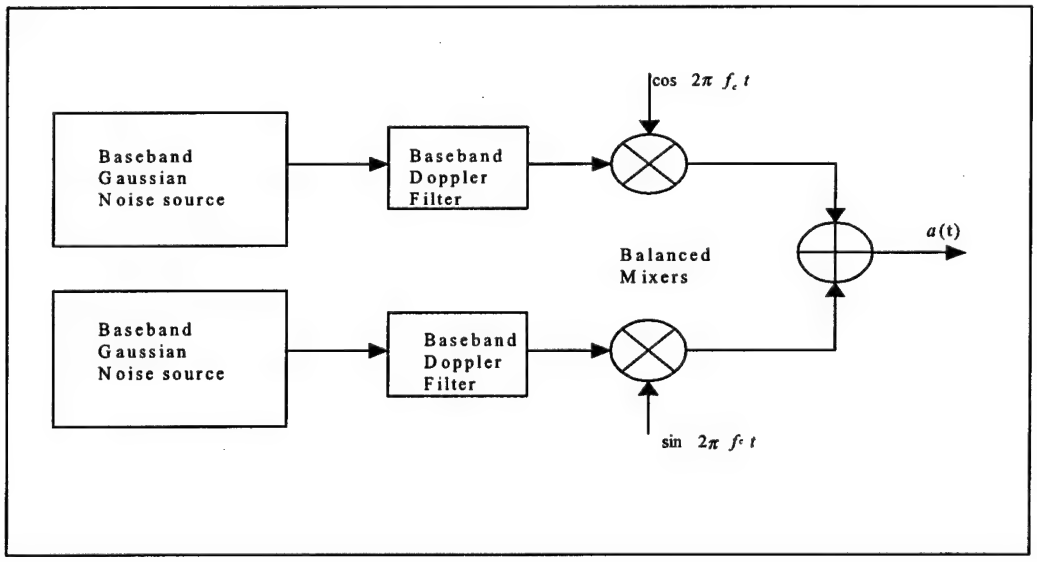

Figure 4-8 Implementation of Rayleigh fading simulator at baseband [14].

The steps used to implement the simulation model shown in Figure 4-8 are as follows [14]:

1. Specify the number of frequency domain points, N, used to represent $\sqrt{S(f)}$ and the maximum Doppler frequency shift f_d. The value used for N is usually a power of 2.

2. Compute the frequency spacing between adjacent spectral lines as $\Delta f = 2f_d/(N-1)$. This defines the time duration of a fading waveform, $T=1/\Delta f$.

3. Generate complex Gaussian random variable for each of the $N/2$ positive frequency components of the noise source.

4. Construct the negative frequency components of the noise source by conjugating positive frequency values and assigning these at negative frequency values.

5. Multiply the in-phase and quadrature noise sources by the fading spectrum $\sqrt{S(f)}$.

6. Perform an inverse fast Fourier transform on the resulting frequency domain signals from the in-phase and quadrature arms to get two N-length time series, and add the squares of each signal point in time to create an N-point time series like under the radical of Equation 4-14.

7. Take the square root of the sum obtained in Step 6 to obtain an N-point time series of a simulated Rayleigh fading signal with the proper Doppler spread and time correlation.

For the discrete model implemented, sampling of the fading process is done at the symbol rate $1/T_s$, of the link. Therefore, flat fading channels are often characterized by a normalized spectrum based on the Doppler bandwidth-symbol duration products, f_dT_s. For example, a cellular link sending 10K symbols/sec through a fading channel with a Doppler bandwidth, f_d = 20 Hz would have f_dT_s = 0.002. Figures 4-9 and 4-10 show sample functions of fading process generated using the model with values of f_dT_s = 0.01 and 0.1 respectively. Note that $f_dT_s << 1$ for the flat fading conditions.

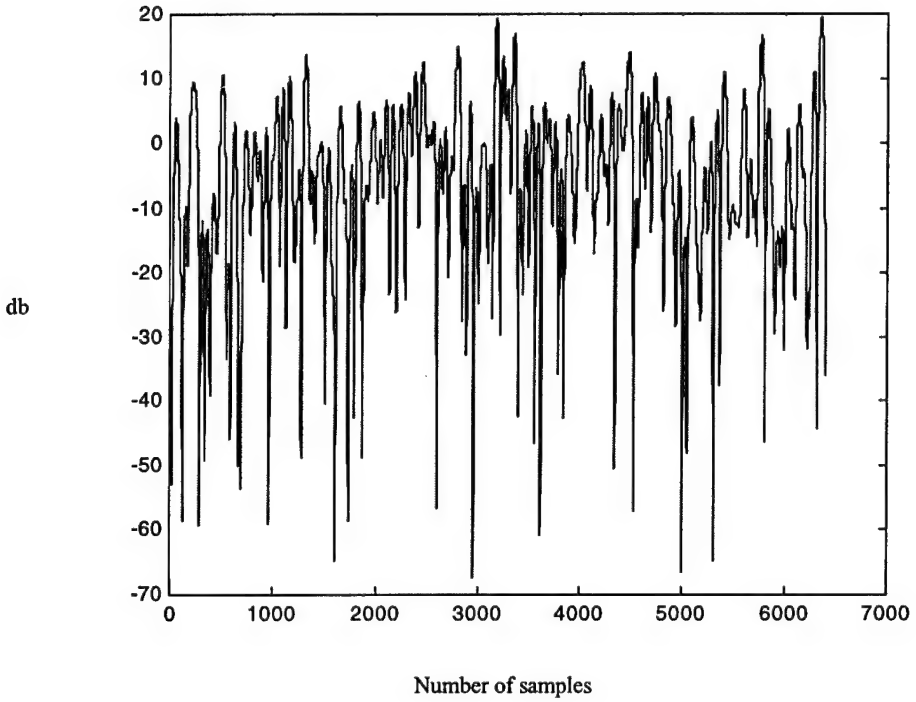

Figure 4-9 Sample of simulated signal envelope waveforms (mean power = 0 db and f_dT_s = 0.01).

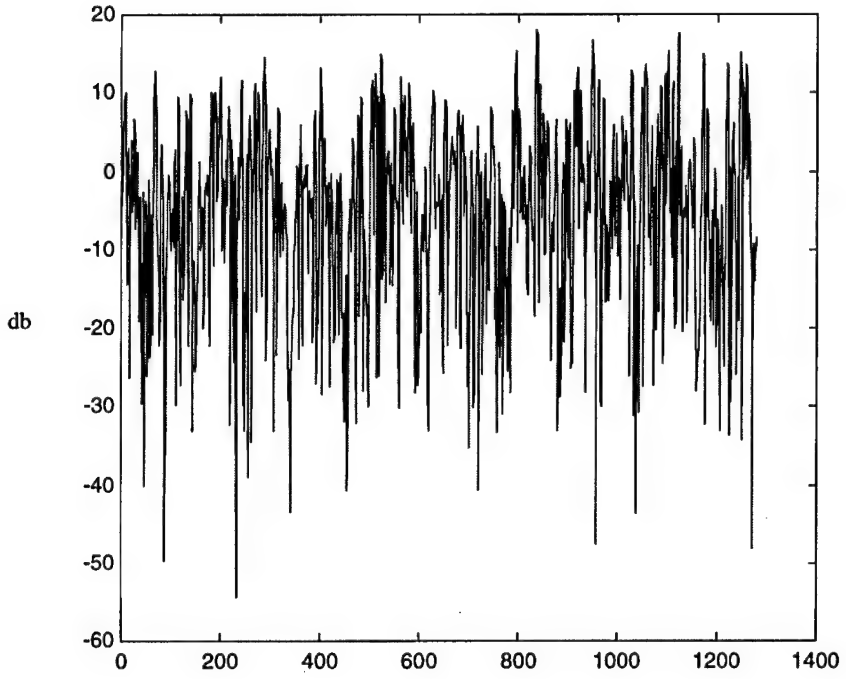

Figure 4-10 Sample of simulated signal envelope waveforms (mean power = 0 db and $f_d T_s = 0.1$).

This model is constructed in the frequency domain using the shaping filters (Doppler filter) to get the required correlation between the adjacent samples. To validate this model, Figure 4-11 shows the normalized autocorrelation of the output fading amplitudes compared with the analytic expression of this autocorrelation given in Equation 4-16.

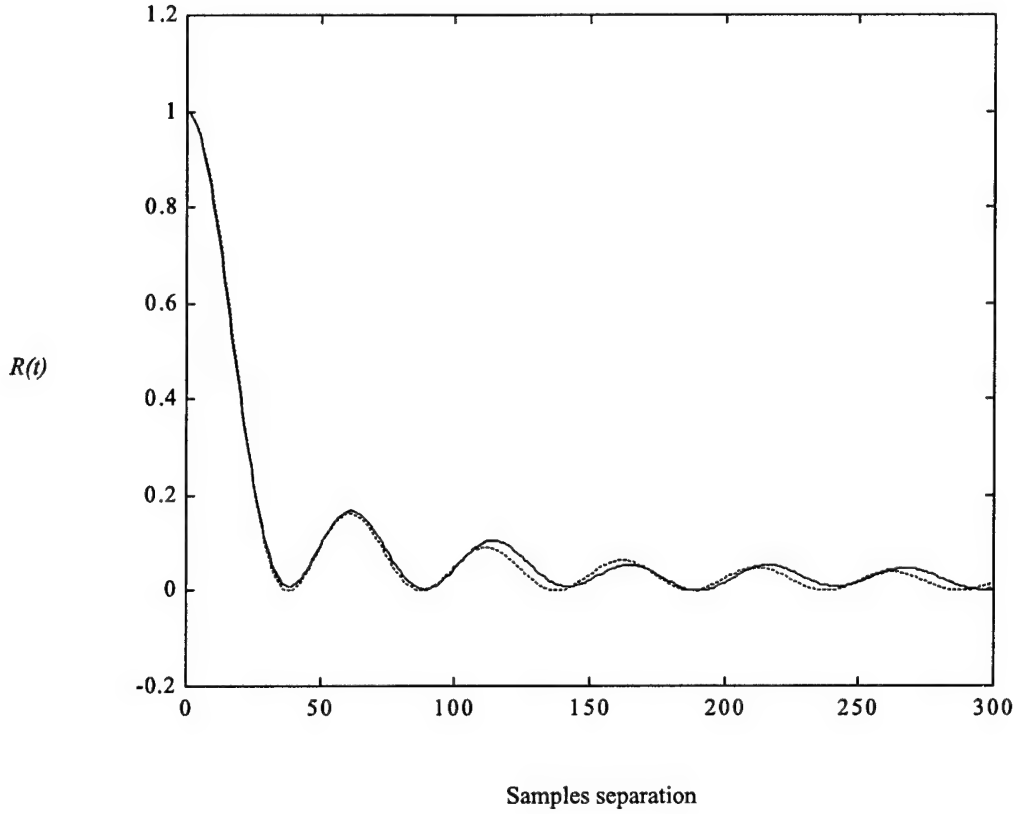

Figure 4-11 Comparison of simulated fades autocorrelation and the analytic autocorrelation.

4.3.2 SOVA Modeling

The Soft-Output Viterbi Algorithm (SOVA) accepts soft inputs of *a priori* information and soft channel values, and produces the reliability of the estimated bits. The SOVA can be implemented in the trace back mode. The classical Viterbi Algorithm (VA) proceeds in the usual way by calculating the metrics for the *m*th path through the trellis using Equation 3-79 with or without *a priori* information. For any state, *s*, and

time i it selects the path with the larger metric $M_{i,s}^{(m')}$, and saves the metric difference $\Delta_{i,s} = M_{i,s}^{(m')} - M_{i,s}^{(m)}$. Using the metric difference $\Delta_{i,s}$ the SOVA updates the reliability of the estimated bits. Then, the SOVA algorithm performs the classical Viterbi algorithm, with minor modification, in a forward process followed by a trace back process that computes the bit reliabilities.

The SOVA algorithm can be summarized as follows:

1. Assume a trellis with 2^{k-1} states, where k is the constraint length of the convolutional encoder, and a frame with length of L bits.

2. The trellis of this encoder consists of $L2^{k-1}$ nodes. Assign to each node the variables $M_{i,s}$ and $\Delta_{i,s}$ which represent the accumulated branch metric and the difference between the competitive metrics at time i and state s, respectively. Also, assign to each node in the trellis the variable $r_{i,s}$ which represents the reliability of the information bit at this node.

3. Initialize all $M_{i,s}$'s with $-\infty$ except the one at time i = 1 and the zero state (assuming start encoding from zero state). Initialize all $\Delta_{i,s,}$'s that contain the metric difference between the competitive paths with any arbitrary values. Also, initialize all the reliability values $r_{i,s}$ to ∞.

4. Starting at time i, initializing with i = 1, and starting from the state zero, compute the branch metric, added to the accumulated metric at the previous state according to Equation 3-79.

5. At each state, for the binary case, there are two branches entering this state. Compare both metrics and selects the larger one and assign it $M_{i,s}$ of that state and save the difference in $\Delta_{i,s}$ of the same state. Save both the survived path information bits and the nonsurvived path information bits.

6. Compare the survived and nonsurvived bits in the previous step. For all the positions where the bits are different, the survived bits reliability need to be updated according to the old value and the value of the metric difference at this node, $\Delta_{i,s}$. Choose the smaller value.

7. Do step 5 and 6 for all the states at time i.

8. Put the time $i = i + 1$, and repeat steps 5, 6 and 7 until the end of the trellis.

9. At the end of the trellis, the survived path is determined and the corresponding information bits with its reliabilities are available.

The forward stage of the SOVA is implemented using the classical Viterbi algorithm with a slight necessary modification [63]. When state sequences are very long or infinite, it is necessary to truncate survivors to manageable lengths δ. In other words, the algorithm must come to a definite decision on nodes up to time i at time $i + \delta$. If the truncation depth δ is chosen large enough, there is a high probability that all time $i + \delta$ survivors will go through the same nodes up to time i. So the initial segment of the maximum-likelihood path is known up to time i and is the output of the algorithm's decision. In this case, truncation costs nothing.

4.3.3 Hypothesis on The Simulation Model

The proposed turbo coded system was simulated for punctured and unpunctured encoded bits. The system used Binary Phase Shift Key (BPSK) modulation. Two types of channels were considered: Additive White Gaussian Noise (AWGN) channel and Rayleigh flat-fading channel. For the Rayleigh flat-fading channel, it was assumed that sufficient channel interleaving existed such that the fading amplitudes were independent from symbol-to-symbol. Without the channel interleaver, the fading was assumed to be correlated.

For the experiments (Monte Carlo simulations) a turbo code composed of two constituent encoders of rate 1/2 and constraint length of $k = 3$ with octal generators $(5/7)_8$ was used. Both encoders were terminated. The decoder was implemented using the SOVA algorithm with 8 iterations furnished with side information (perfect channel estimation).

All the experiments were conducted over a wide range of signal-to-noise ratios, E_b/N_0. At least 5000 errors were accumulated for the lowest signal-to-noise ratio E_b/N_0 ≤ 2 db, and 200 errors were accumulated for $E_b/N_0 > 2$ db. At high signal-to-noise ratios, the simulation becomes quite lengthy. It was found that 200 errors is sufficient for reliable results [95, 96].

4.4 Summary

This chapter developed the analytical models needed to evaluate the analytical bound. The idea of the uniform interleaver that makes the calculation of the analytical

bound tractable was presented in detail. A method of calculating the average weight enumerating function of turbo codes through the calculation of the weight enumerating function of the convolutional codes that used as a constituent decoder of the parallel concatenation was presented.

The second half of this chapter presented the development of the simulation models of turbo codes, and channel model used in this thesis along with its implementation issues. The Soft Output Viterbi Algorithm (SOVA) algorithm was discussed. The use of the SOVA as a constituent decoder of the turbo code decoder, along with practical implementation issues was presented.

5 Performance Enhancement of Turbo Codes with Short Frames

5.1 Introduction

In wireless communication systems such as cellular systems, information is typically sent in short frames (less than 300 bits). The size of the transmission frame limits the choice of error control codes. Convolutional codes are commonly employed in wireless systems. One of the advantages of convolutional codes is that the performance is independent of the frame size as long as the frame size is much larger than the constraint length of the code.

Turbo codes have been shown to approach the Shannon limit for error correcting capability at low signal-to-noise ratios given large sized frames. Extensive research efforts have been examining ways to enhance the performance of turbo codes for short frames (e.g., voice transmission).

Section 5.2 presents one way of enhancing the performance by optimizing the energy allocated to each bitstream to achieve the best performance possible. In standard turbo codes, all bits are transmitted with equal energy. Changing the energy allocation strategy can enhance the performance of turbo codes with short frame lengths.

Section 5.3 presents another way of enhancing the performance of turbo codes with short frames by proper design of the interleaver. With proper design of the interleaver, the pairing of low-weight sequences to the input of the encoders can be avoided. Circular shift interleavers can be used to ensure that the minimum distance due to weight-2 input sequences grows roughly as $\sqrt{2N}$, where N is the block length. The generalization of the mapping function of the circular shift interleaver to be suitable for both equal and unequal error protections is presented. Section 5.4 summarizes the contents of this chapter.

5.2 Energy Allocation Strategies

The output of turbo codes (shown in Figure 5-1) has three bitstreams to be multiplexed. One bitstream represents the systematic bits and the other two represent the checking bits of the first and second encoders. One way of enhancing the performance is to optimize the energy allocated to each bitstream to achieve the best performance possible. In the standard turbo codes, all systematic and parity bits are transmitted with equal energy. This energy allocation does not guarantee optimum performance.

In [97, 98], two different strategies for allocating the energy are presented. In [97], the authors investigated improvement of the performance at very low signal-to-noise ratios (within 0.5 db of the Shannon limit). They conclude that by allocating more energy to the systematic bits, better performance can be achieved. At these very low signal-to-noise ratios, the bit error rate is not practical for most applications (e.g., voice transmission). In [98], the authors investigated optimizing the energy allocation at higher signal-to-noise ratios. They concluded that at larger signal-to-noise ratios, the fraction of

the total energy allocated to systematic bits is usually lower than that of the parity bits. The authors' results were supported for frame lengths of 1000 bits.

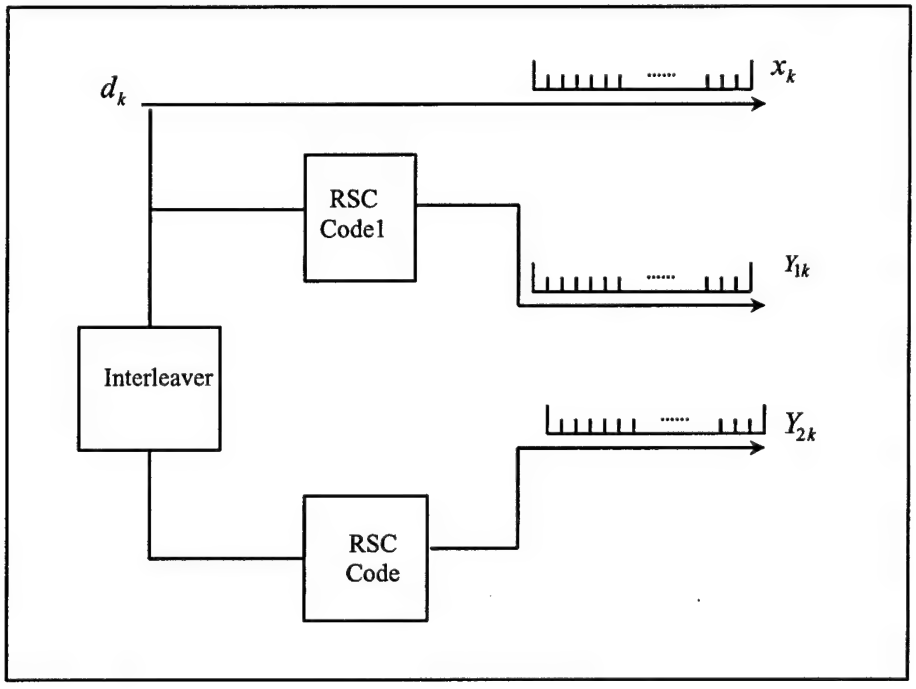

Figure 5-1 Block diagram of turbo-encoder with three output bitstreams.

5.2.1 System Model

The system model used in the simulation is based on a turbo code with two identical parallel encoders. Each encoder uses octal generators $(5/7)_8$. Both encoders are terminated using the algorithm given in [38]. The encoded bits, without puncturing, (rate 1/3) are modulated using a Binary Phase Shift keying (BPSK) modulator and transmitted over an Additive White Gaussian Noise (AWGN) channel. A random interleaver is also used. The decoder is composed of two component decoders, each implemented by the

Soft Output Viterbi Algorithm (SOVA) [51] with 8 iterations (note that in the previous papers [97, 98], the authors use the MAP [2] or modified MAP [32] algorithms). The decoding algorithm is modified to handle the unequal distribution of the total energy. This modification comes at no additional complexity.

5.2.2 Bound Modification

Assuming the maximum-likelihood decoding, Binary Phase Shift Keying (BPSK) modulation used, and an Additive White Gaussian Noise (AWGN) channel, the union bound is given by [61, 62]:

$$P_b \le \sum_j \sum_w \frac{w}{N} A_{w,j}^{C_p} P_2(j) \tag{5-1}$$

where N is the frame length and $A_{w,j}^{C_p}$ denotes the number of codewords of the parallel concatenation (turbo code) with weight j generated by a word of information with weight w. Parameters d_1 and d_2 represent the parity weights of the first and second encoders respectively. The total weight, j, is $j=d_1+d_2+w$. $P_2(j)$ represents the pair-wise error probability between the zero-codeword and the codeword with weight j. In the case of an AWGN channel, $P_2(j)$ is given by [12]:

$$P_2(j) = Q\left(\sqrt{2jr\frac{E_b}{N_0}}\right) \tag{5-2}$$

where r is the code rate of the code, E_b/N_0 is the signal-to-noise ratio per information bit, and $Q(*)$ is the tail integral of a standard Gaussian density with zero mean and unit variance.

Investigation of the performance of the turbo code bound with unequal energy distribution between the systematic bits and the parity bits can be viewed as changing the signal-to-noise ratio of the systematic bitstream while keeping the average signal-to-noise ratio E_b/N_0 constant. If the total energy allocated per information bit is E_b, then let the energy allocated to the systematic bit be E_s and the energy allocated to both parity bits be E_{p1} and E_{p2}, such that Equation 5-3 is satisfied:

$$E_b = E_s + E_{p_1} + E_{p_2} \tag{5-3}$$

In standard turbo codes (with equal energy distribution), Equation 5-4 is also satisfied.

$$E_s/E_b = E_{p_1}/E_b = E_{p_2}/E_b = 1/3 \tag{5-4}$$

Denoting the ratio of the portion of the energy allocated to the systematic bit, E_s, to the total energy per information bit, E_b, by $c=E_s/E_b$, then $E_s=cE_b$ and the energy allocated to each parity bit is $E_{p1}=E_{p2}=(1-c)E_b/2$. For a codeword with weight j generated from a word of information weight w, $(j-w)$ parity bits are produced. In the standard turbo codes, the squared Euclidean distances between the zero-codeword and the codeword with weight j is rjE_b.

Now, the Euclidean distance must be modified to accommodate for the unequal distribution of the energy. For each frame, there are w information bits each with energy $E_s=cE_b$ and$(j-w)$ parity bits each with energy $E_{p1}=E_{p2}=(1-c)E_b/2$. The modified squared Euclidean distances can be written as $(wc+(j-w)(1-c)/2)E_b$. Substituting in the pair-wise error probability in Equation 5-2, and into Equation 5-1, the probability of bit error becomes:

$$P_b \le \sum_j \sum_w \frac{w}{N} A_{w,j}^{c_p} \, Q\left(\sqrt{\frac{(2wc + (j-w)(1-c)E_b}{N_0}}\right) \tag{5-5}$$

Note that if $c = 1/3$, this bound yields to the standard turbo code.

5.2.3 Simulation and Analytical Results

Using the simulation model described in Section 5.2.1, and modified analytical bounds, the performance of turbo codes is investigated. This investigation uses the SOVA decoder, with two frame lengths of 48 and 192 bits over a wide range of signal-to-noise ratios. Since the analytical bounds diverged at lower signal-to-noise ratios, the simulation model is used to predict the performance of turbo codes with unequal energy distribution at lower signal-to-noise ratios.

By changing the power allocated to each bitstream, there is an improvement in the performance of turbo codes. Tables 5-1 and 5-2 show the comparison between the bit error rate (BER) for standard turbo codes with equal energy distribution and the minimum bit error rate occurring at certain energy distributions at the same signal-to-noise ratio, E_b/N_0, for frame lengths of 48 and 192 bits respectively. For example, in Table 5-1, for a frame length of 48 bits, the bit error rate is 1×10^{-1} at 0 db for the standard turbo code with the equal energy distribution (each bitstream has 1/3 of E_b/N_0). This energy distribution is denoted in the table by (1/3, 1/3, 1/3). At the same signal-to-noise ratio, the minimum bit error rate is 6.9×10^{-2} with $0.7E_b$ allocated to the systematic bit and 0.15 E_b allocated to each parity bit. This energy distribution is denoted in Table 5-1 by (0.7,0.15, 0.15) for the unequal energy distribution.

Figures 5-2 and 5-3 show the detailed simulation results of the simulated bit error rate (BER) for two signal-to-noise ratios, at different E_s/E_b values for 48 and 192 bits frames respectively. At a very low signal-to-noise ratio (0 db in Figure 5-2 and – 0.5 db in Figure 5-3), the bit error rate decreased as the fraction of the energy allocated to the systematic part increases.

Table 5-1 Comparison of Equal and Unequal energy distribution for 48-bit frames turbo codes.

SIGNAL-TO-NOISE RATIO, E_b/N_0	STANDARD TURBO CODE (EQUAL ENERGY DISTRIBUTION)		MODIFIED TURBO CODE (UNEQUAL ENERGY DISTRIBUTION)	
	BER	Distribution	BER	Distribution
0 db	1.1×10^{-1}	(1/3, 1/3, 1/3),	6.9×10^{-2}	(0.7,0.15,0.15)
3.0 db	1.7×10^{-3}	(1/3, 1/3, 1/3),	7.7×10^{-4}	(0.4,0.3,0.3)

Table 5-2 Comparison of Equal and Unequal energy distribution for 192-bit frames turbo codes.

SIGNAL-TO-NOISE RATIO, E_b/N_0	STANDARD TURBO CODE (EQUAL ENERGY DISTRIBUTION)		MODIFIED TURBO CODE (UNEQUAL ENERGY DISTRIBUTION)	
	BER	Distribution	BER	Distribution
0 db	1.1×10^{-1}	(1/3, 1/3, 1/3),	8.9×10^{-2}	(0.9,0.09,0.01)
1.5 db	4.7×10^{-3}	(1/3, 1/3, 1/3),	4.4×10^{-3}	(0.5,0.25,0.25)

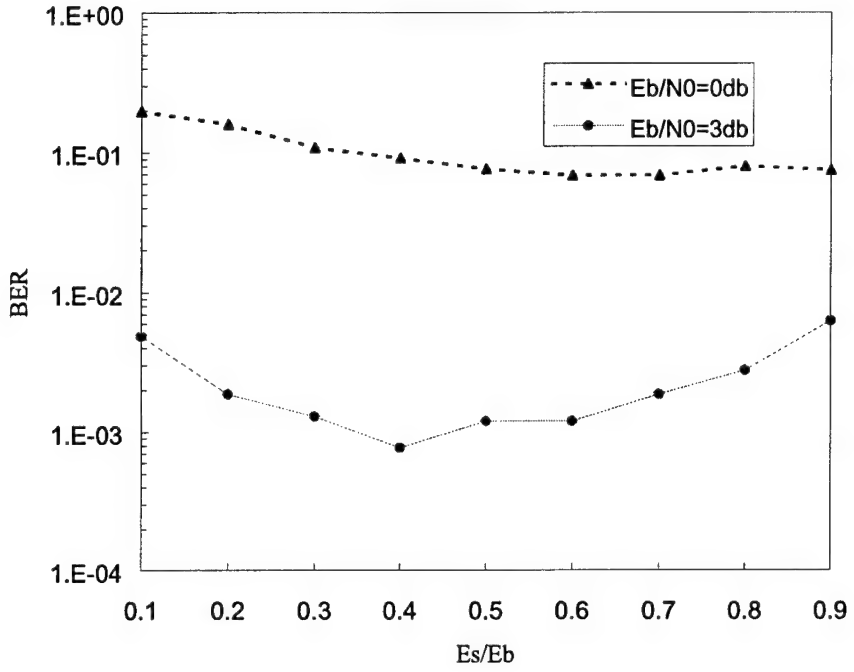

Figure 5-2 Simulated turbo code with frame lengths of 48 bits at two signal-to-noise ratios.

Similar to [97], at very low signal-to-noise ratios, the performance is enhanced by allocating more energy to systematic bits. At practical signal-to-noise ratios around BER=10^{-3}, the typical error rate required for voice transmissions are between 5×10^{-3} and 4×10^{-2}. Also, the simulated bit error rate at a signal-to-noise ratio of 3 db (for 48 bits frames) is shown in Figure 5-2 and of 1.5 db (for 192 bits frames) in Figure 5-3. Figures 5-2 and 5-3 show that the minimum BER occurs at E_s/E_b = 0.4 i.e., (0.4, 0.3, 0.3) in Figure 5-2 and in Figure 5-3 at E_s/E_b = 0.5. Any increases in E_s/E_b causes degraded performance. Therefore, limitations exist in increasing the energy allocated to the systematic bits operating in this region.

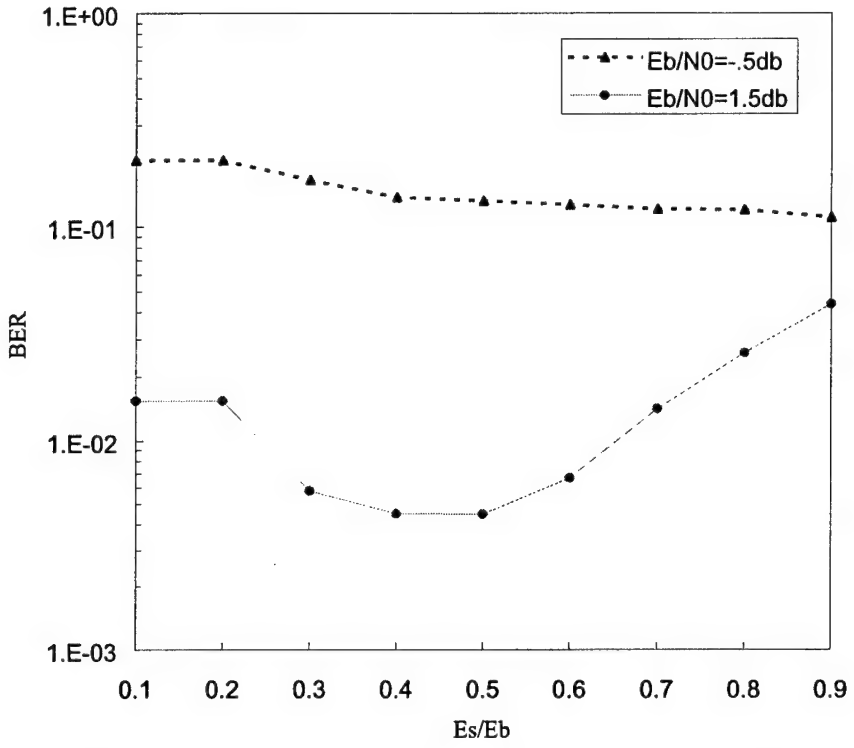

Figure 5-3 Simulated turbo code with frame lengths of 192 bits at two signal-to-noise ratios.

At higher signal-to-noise ratios, the modified analytical bound given in Equation 5-5 is used to handle the unequal energy distribution. Figures 5-4 and 5-5 show the analytical bound on BER for higher signal-to-noise ratios at different values of E_s/E_b using 48 and 192 bits frames respectively. Figures 5-4 and 5-5 show that reducing the energy allocated to the systematic bits improves the performance up to a point where any reduction less than $E_s/E_b = 1/3$ will not improve the performance. As shown in Figure 5-4, this occurs at $E_b/N_0 = 4.0$ db and in Figure 5-5 this occurs at $E_b/N_0 = 5.5$ db.

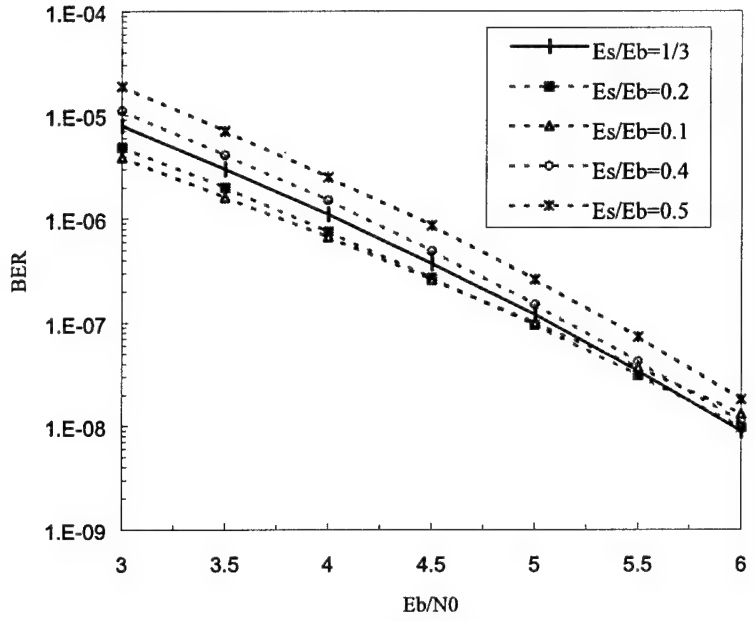

Figure 5-4 Modified bounds of turbo code with frame length of 48 bits.

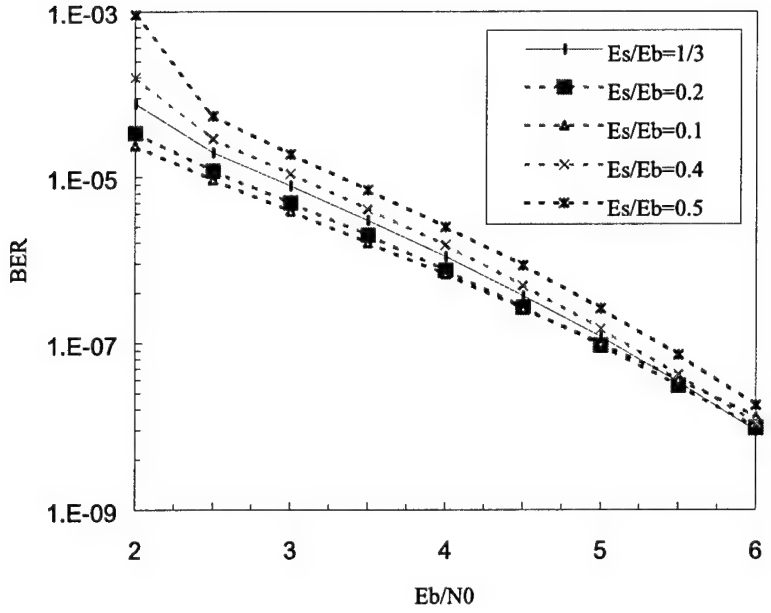

Figure 5-5 Modified bounds of turbo code with frame length of 192 bits.

5.3 *General Interleaver for Equal and Unequal Error protection*

Key to the efficient operation of turbo codes is the interleaver design. With proper design, the pairing of low-weight sequences to the input of the encoders can be avoided. Circular shift interleavers can be used to ensure that the minimum distance due to weight-2 input sequences grows roughly as $\sqrt{2N}$, where N is the block length. For large frame lengths, an interleaver that permutes the data in a random fashion provides better error performance than structural interleavers. For short frame lengths, structural interleavers can perform as well as any random interleaver.

In many speech and image coding schemes, certain coded information bits have higher sensitivity to channel errors than the other bits. In order to maximize channel use, unequal error protection codes are needed. Turbo codes that use Unequal Error Protection (UEP) have been introduced in previous research [80]. This research reported that UEP turbo codes could not be achieved by puncturing low rate codes alone if the information symbols corresponding to different classes are spread over the interleaved frame. It was also noted that the performance would be close to that of an Equal Error Protection (EEP) code. To overcome this problem, the authors in [80] proposed using a different interleaver for each protection level. This research uses only one interleaver to accomplish the unequal error protection required.

5.3.1 *Circular Shift Interleaver*

The performance of turbo codes depends on the type and size of the interleaver used. For short frame sizes, a structural interleaver can perform as well as any random interleaver that uses the same frame length [66]. Structural interleaving based on circular

shifting has been discussed in [41] as one of nonrandom interleavers. A circular shift interleaver is governed by the following formula:

$$\Pi(j)=(aj+r) \mod N, \quad j=0,1,...,N-1 \tag{5-6}$$

where j is the bit position inside the frame before interleaving, $\Pi(j)$ is the interleaved position of bit j after interleaving, N is the interleaver length, $r < N$ an offset, $r = 0$ if the edge effect is ignored, and $a < N$ is a step size that is relatively prime to N. For illustration, Table 5-3 shows, the interleaving mapping generated by this formula with $N = 18$, $a = 5$, and $r = 0$.

Table 5-3 Interleaving map for circular shift interleave ($N = 18$, $a = 5$, $r = 0$).

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\Pi(j)$	0	5	10	15	2	7	12	17	4	9	14	1	6	11	16	3	8	13

In this example, if the distance between a pair of 1's in an uninterleaved weight-2 input sequence is $d_1 = 1$, then the corresponding distance after interleaving is either $d_2 = 5$ or $d_2 = 13$. Similarly, if $d_1 = 2$ then either d_2=10 or $d_2 = 8$. This ensures that d_1+$d_2 \geq 6$ for any possible combination of d_1 and d_2. If N is half of a perfect square, and taking the step size $a = \sqrt{2N} - 1$, then $d_1 + d_2 \geq \sqrt{2N}$ is guaranteed for all possible combinations of d_1 and d_2. The step size a that achieves this inequality is not unique. For example, $a = \ell\sqrt{2N} \pm 1$ also gives the same lower bound on d_1+d_2 for positive integers $\ell < \sqrt{N/2}$

that are relatively prime to $\sqrt{N/2}$. In the case where $\sqrt{2N}$ is not an integer, $d_1+d_2 \geq \mu$, where μ is slightly smaller than $\sqrt{2N}$, can be achieved for properly optimizing the step size.

Given an application that needs large frame lengths, random interleavers have been shown to outperform structural interleavers. For this reason, circular shift interleavers have been ignored for use in large frame lengths. The performance of the circular shift interleaver has not been studied before using turbo codes with short frame lengths. A first-of-its-kind performance comparison of turbo codes using block, random, and circular shift interleavers is presented here. Figure 5-6 shows the performance of the three interleavers.

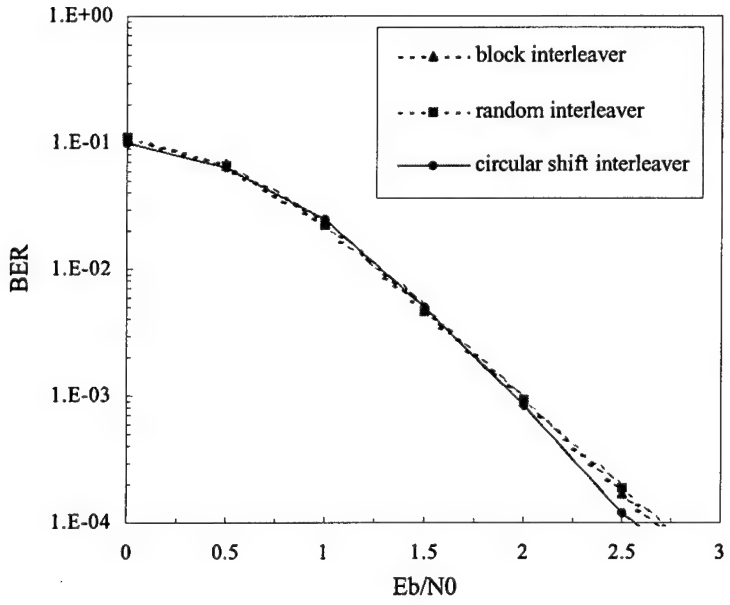

Figure 5-6 Performance of different turbo code interleavers for 192-bit frame lengths.

In Figure 5-6, the performance of the three interleavers is approximately the same up to 10^{-4} BER for short frame lengths. At a BER of 10^{-4}, the performance of the circular shift interleaver is 0.3 db better than the other two interleavers. From this, it is concluded that the circular shift interleaver is suitable for short frame lengths. As long as a is an odd number and r is even or zero, this interleaver is an odd-even interleaver. This type of interleaver has been shown to have advantages for turbo code puncturing [69], yielding uniform distribution of the correcting capability of the codes in each dimension.

5.3.2 A General Circular Shift Interleaver

Problems associated with combining different rates in the same frame have been reported in [80]. Here, the authors proposed a method to avoid degradation in performance and still have the freedom to choose the necessary rates. This method is based on using different interleavers for each protection level. Using this strategy, the outputs of different interleavers can be connected in series. This research uses only one modified circular shift interleaver.

Let the input frame length to the interleaver be N with M subframe protection levels, each of them with length N_m bits, $1 \le m \le M$ as shown in Figure 5-7. In Figure 5-7, L_m, $1 \le m \le M$ represents the accumulated length of the preceding level lengths before the level m. Intuitively $L_1 = 0$, and the other L_m are defined as follows:

$$L_m = \sum_{i=1}^{M-1} N_i , \qquad 2 \le m \le M \tag{5-7}$$

Now, the general interleaver is defined as follows:

$$\Pi(j) = \sum_{m=1}^{M} b_m \Pi_m(j) \tag{5-8}$$

where,
$$b_m = \begin{cases} 1 & \text{if } L_m < j \le L_{m+1} \\ 0 & \text{Otherwise} \end{cases} \tag{5-9}$$

and $\Pi_m(j)$ represents the interleaving map within the subframe level m, and is given by

$$\Pi_m(j) = L_m + \{a_m(j - L_m) \mod N_m\}, \qquad 1 \le m \le M \tag{5-10}$$

In Equation 5-10, $a_m < N_m$ is a relatively prime to N_m and $(j\text{-}L_m)$ represents the position of bit j in the subframe m. By employing the generalized mapping function defined in Equation 5-8, the whole frame can be interleaved by keeping each level separated from other levels using the our generalized circular shift interleaver.

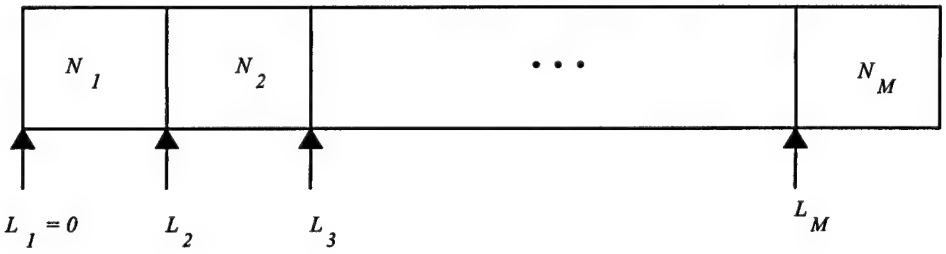

Figure 5-7 Frame of length N with M-level error protection each with length N_i and starting at bit number L_i.

Note that the circular shift interleaver defined in Section 5.3.1 for equal error protection is a special case of the generalized one by letting $M = 1$ in Equation 5-8. In

Table 5-4, the interleaving mapping is generated using the generalized interleaver for two level of protection, i. e. $M = 2$, with input frame length $N = 32$ and level 1 and 2 lengths are 12 and 20 bits respectively. In this example, $a_1 = a_2 = 7$.

5.3.3 Simulation Results

The system model used in the simulation is based on the turbo code with two identical parallel concatenated Recursive Systematic Convolutional constituent encoders. Each encoder uses octal generators 5 (feedforward) and 7 (feedback). Both encoders are terminated using the scheme in [38]. The encoded bits are modulated using a Binary Phase Shift Keying modulator and transmitted over an Additive White Gaussian Noise channel. The decoding is performed using a Soft Output Viterbi Algorithm constituent decoder [51] with 8 iterations.

The simulation model uses a frame length of 192 bits which consists of 190 information bits plus 2 bits for termination. Only 190 bits need to be interleaved. Three levels of error protections have been implemented within each frame. Table 5-4 illustrates the length and the rate for each level.

The information bits are ordered inside the frame in decreasing order of importance. This strategy allows for the greatest performance benefits. The results from the EEP simulation show that the bits at the beginning of the frame have lower error rates than the bits at the end of the frame.

Three different puncturing matrices and one modified circular shift interleaver have been modeled. The systematic portion of the code is not punctured. The model parameters are set to: $a_1 = a_2 = 7$, $a_3 = 13$, $r_1 = 0$, and $r_2 = r_3 = 10$.

Table 5-4 Interleaving map for modified circular shift interleave ($N = 32$, $N_1 = 12$, $N_2 = 20$).

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\Pi(j)$	0	7	2	9	4	11	6	1	8	3	10	5	16	23	30	17
	24	31	18	25	12	19	26	13	20	27	14	21	28	15	22	29

Table 5-5 192-bit frame with 3 levels of local coding rates.

Level	Code rate	Information bits	Check + information bits
1	1/3	32	96
2	2/5	48	120
3	2/3	112	168
Overall frame	1/2	192	384

Figure 5-8 shows the short term (8 bits) bit error rate (BER) with three-level UEP. In this example, the code rate is changing from 1/3 to 2/5 to 2/3 at an $E_b/N_0 = 0.5$ db. It is shown in Figure 5-8 that there is no "spill over" effect when switching from one level to another. Additionally, each rate provides the expected error probability. The transition regions are comparable to those presented in [80]. An UEP turbo code is also compared with an EEP turbo code with the same frame length and same overall code rate of 1/2. Figure 5-9 shows the BER performance versus E_b/N_0 for levels 1, 2, and 3 of the UEP

scheme with respect to the EEP scheme. For example, to achieve a BER $\leq 10^{-3}$, 10^{-2} and 0.15, for the three UEP levels, respectively, the UEP turbo code meets these conditions at $E_b/N_0 = 1.1$ db, while the EEP turbo code $E_b/N_0 = 2.4$ db.

As in the case of equal error protection, the proper selection of a guarantees that the separation between the bits in the same level will grow with $\sqrt{2N_m}$. Also, the separation between the level edges can be controlled by adding an offset term $r_m < N_m$ to Equation 5-10, as shown in Equation 5-11:

$$\Pi_m(j) = L_m + \{[a_m(j - L_m) + r_m] \mod N_m\}, \qquad 1 \leq m \leq M \tag{5-11}$$

5.4 Summary

In the first part of this chapter, the problem of energy allocations in turbo codes was examined. In standard turbo codes, all bits are transmitted with equal energy. This strategy does not guarantee optimum allocation. For turbo codes with short frames, different ways to allocate the energy was investigated using computer simulation and analytic bounds. It was shown that, for turbo codes with short frames (we use the SOVA as a constituent component of the decoder) operating in very low signal-to-noise environments, allocating more (without any restriction on the amount of increase) energy to the systematic bits improves performance. At higher signal-to-noise ratios, allocating less energy to the systematic bits improves the performance. The most important results are seen at BER around 10^{-3}. The results in this region show that by allocating half of the energy to the systematic bits, the best performance can be achieved. Any deviation from this half energy point results in degraded performance.

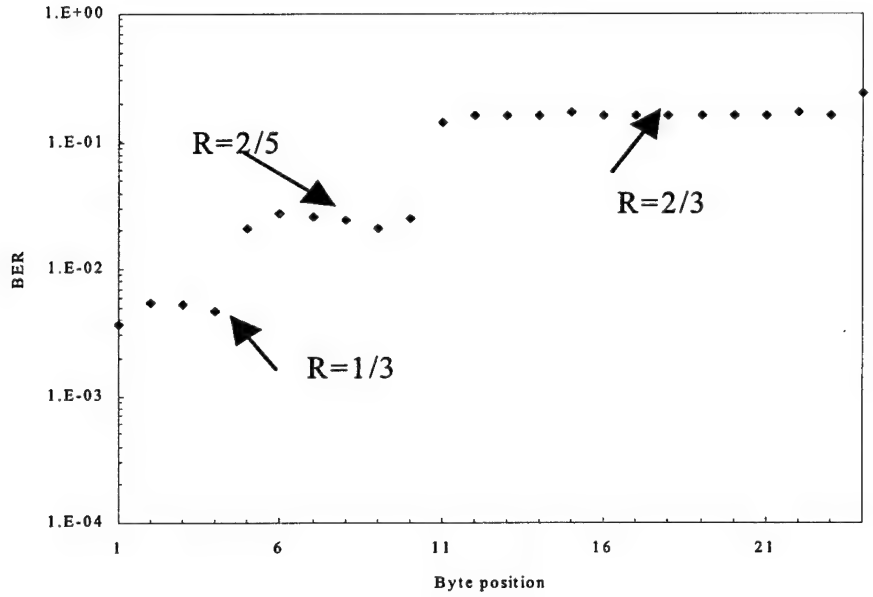

Figure 5-8 Short term BER for three-level Interleaving at $E_b/N_0 = 0.5$ db.

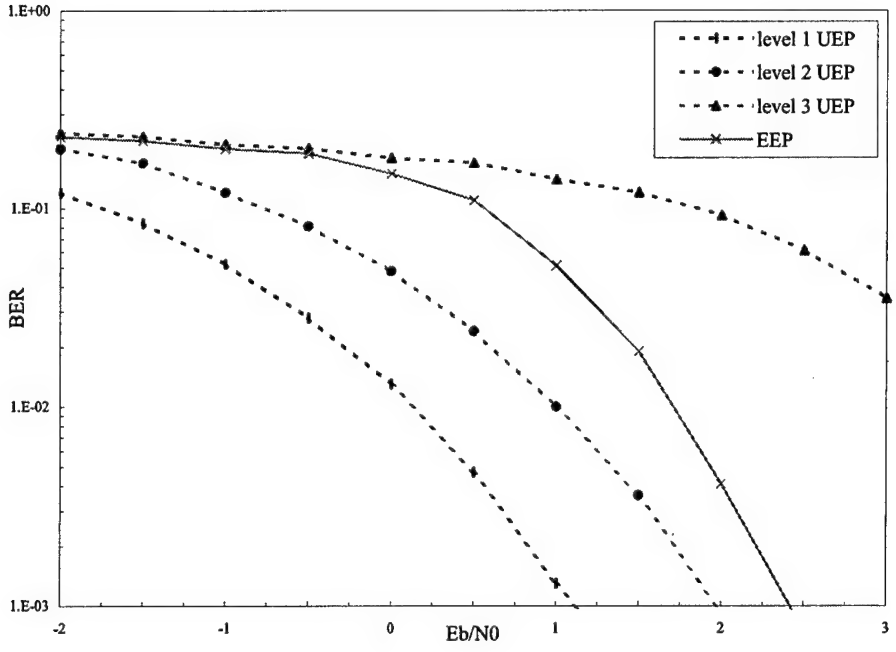

Figure 5-9 BER performance comparison of EEP and UEP turbo code.

The latter portion of this chapter proposed and examined the application of a circular shift interleaver and its generalized version for both equal and unequal error protections with turbo codes for short frame lengths. In the case of equal error protection, the circular shift interleaver has slightly better performance at higher E_b/N_0 compared with block and random interleavers. In the case of unequal error protection, the generalized circular shift interleaver offers a comparable performance to those using different inteleaver for each level of protection.

6 Performance Bounds of Punctured Turbo Codes

6.1 Introduction

In this chapter, the average upper bound of punctured turbo codes is derived from the knowledge of the conditional weight enumerating function of the original low-rate code. A novel approach for generating these bounds, *hypergeometric puncturing,* is introduced to average the performance over all possible punctured positions at a required code rate. From this, an analytic formula for error performance is derived. Values obtained from simulation trials are shown to be convergent with analytic results. Simulation results in Additive White Gaussian Noise (AWGN), fully interleaved fading, and correlated fading channels are also presented along with the analytical bound.

Turbo codes [2] have been shown to achieve near-Shannon-limit error correction performance with relatively simple component codes and large interleavers. For a bit error probability of 10^{-5} and code rate = 1/2, it has been shown that an E_b/N_0 of 0.7 dB is required for block lengths of 65,536 bits. A typical turbo code encoder is shown in Figure 6-1. This encoder consists of two binary rate 1/2 convolutional encoders, an interleaver of length N, along with puncturing and multiplexing devices. Without the puncturing device, the encoding is rate 1/3. The decoding process relies on iterative

processing in which each component decoder takes advantage of the work performed by the other in the previous step.

A punctured turbo code is a high-rate code obtained by the periodic deleting of specific code symbols from the output of a low-rate code. The resulting high-rate code depends on both the low-rate code (original code) and on the number and specific positions of the punctured symbols. Since the first appearance of turbo codes, puncturing has been used to increase the code rate. Studies [56, 84, 99, 100, 101] that focused on punctured turbo codes have relied on simulation due to complexity associated in the analytical modeling.

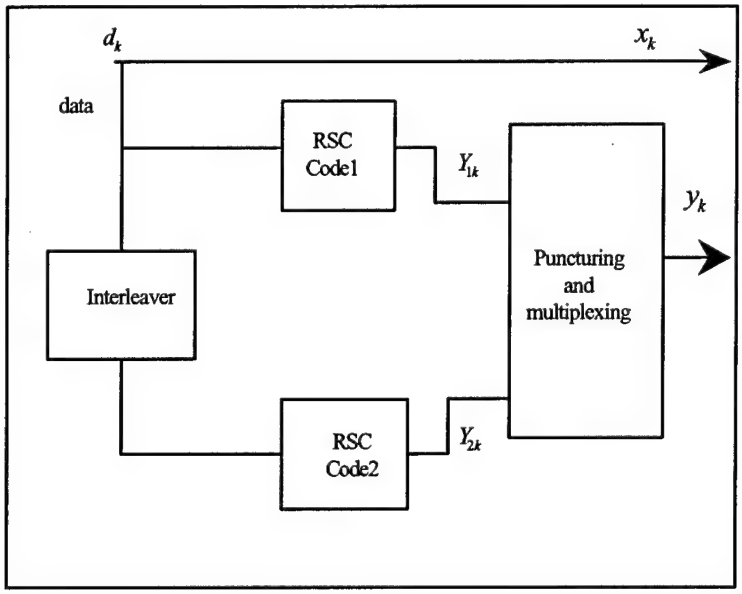

Figure 6-1 Turbo-Code Encoder.

Many of the theoretical and structural properties of turbo codes are discussed in [42, 61, 62, 102, 103]. The most complete works in the analytic bounds are presented in [61, 62]. In [61], the authors derive an analytical upper bound for the average performance of the coding scheme. The average upper bound is constructed by averaging over all possible interleaver configurations. This upper bound is shown to be independent of the interleaver used and reveals the influence of the interleaver length on the code performance. In [62], the authors apply the transfer function bound techniques to obtain an upper bound on the probability of bit error. In their study, the authors developed a method for a recursive computation of the Weight Enumerating Function (WEF) of the convolutional code that is used as a constituent encoder with random interleaving for the calculation of the bound of the turbo code.

This chapter extends the results presented in [61] and [62] by deriving an upper bound of the punctured turbo codes. This is accomplished by averaging over all punctured positions, which in turn yields all possible punctured weights. Section 6.2 presents a background of the punctured convolutional codes. Section 6.3 introduces useful definitions and notations that are used in the derivation of the punctured bound. The derivation of the punctured bound is presented in Section 6.4. In Section 6.5, the punctured bound is applied to AWGN, fully interleaved fading, and correlated fading channels. Section 6.6 evaluates the performance of these bounds and provide a comparison of the analytical bound with simulation results.

6.2 *Punctured Convolutional Codes*

Punctured convolutional codes were first introduced in [104] for the purpose of obtaining simpler Viterbi decoding for rates b/v codes with two branches arriving at each node of the trellis instead of 2^b branches. A punctured convolutional code is a high-rate code obtained by the periodic deleting of specific code symbols from the output of a low-rate code. The resulting high-rate code depends on both the lower-rate code and the number and specific positions of the punctured symbols. The pattern of the punctured symbols is called the perforation pattern of the punctured code.

Consider an original low-rate encoder with rate $1/v_0$. The punctured code with rate b/v can be obtained from the original low-rate code by deleting specific symbols according to the perforation pattern. The perforation pattern can be expressed as a matrix [P] having v_0 rows and b columns with only binary elements 0s and 1s, corresponding to the deleting and keeping of the corresponding code symbols of the original encoder [105, 106]. The perforation pattern contains v 1s. Both the punctured code and its rate can be varied by suitably modifying the elements of the perforation matrix. For example, starting from the original code of rate 1/2, to get rate 2/3 punctured code the following perforation matrix, P1 is used:

$$[P1]=\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

achieving a rate 4/5 can be accomplished using the following perforation matrix, P2:

$$[P2]=\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Variable rate coding can be achieved if all punctured rates of interest are obtained from the same lower-rate code. Only the perforation matrices need to be modified. By adding a restriction that all the code symbols of the higher rate punctured codes are required by the lower rate code, this means that any 1s exist in the higher rate perforation matrix must be exist in the perforation matrix of the lower-rates codes. For example, starting from the original code with rate 1/2, the following three perforation matrices P3, P4, and P5 are satisfying this condition and yielding 4/7, 4/6, and 4/5 punctured rates, respectively:

$$[P3]=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad [P4]=\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad [P5]=\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Punctured codes that satisfying this restriction are said to be rate-compatible [79]. Rate-Compatible Punctured Convolutional (RCPC) codes are useful in some rate adaptive ARQ/FEC applications since only the incremental redundancy needs to be transmitted as the coding rate is decreased. Families of good rate compatible punctured codes have been found in [17, 79].

The error performance of punctured convolutional codes may be evaluated by computing the upper bounds on the bit error probability. Using the transfer function bounding technique, the upper bound of the bit error probability can be evaluated. The transfer function of a convolutional code is evaluated by solving equations describing the transitions between the states of the finite-state encoder. For punctured convolutional codes, the first step in the evaluation is the drawing of a proper state diagram for the encoder under consideration. Reflecting the assumption made on the punctured code trellis [106], knowledge of the perforation matrix is necessary to get the transfer function of the punctured code.

6.3 Analytic Background

It is useful to consider turbo codes as block codes. The input sequences are restricted to length N, where N corresponds to the size of the interleaver in the turbo code encoder. For discussion, consider a traditional union upper bound for maximum likelihood decoding of an (N, $3N$) block code. Without loss of generality, the all-zero codeword is sent, and the upper bound of the probability of word error is [61]:

$$P_e \leq \sum_{d=d_{free}}^{3N} B_d \; P_2(d) \tag{6-1}$$

where B_d denotes the number of codewords with Hamming weight d, d_{free} is the minimum Hamming weight of all the non-zero codewords of the code and it is known as the free distance of the code, and $P_2(d)$ is the pair-wise error probability between the all-zero codeword and codeword with weight d. This bound can be calculated if the Weight Enumerating Function (WEF) [62] of the code, given by Equation 6-2, is available.

$$B^c(H) = \sum_{d=0}^{3N} B_d \; H^d \tag{6-2}$$

In Equation 6-2, H is a dummy variable. Due to the fact that the WEF enumerates the codewords weights without relating them to the input weight of the information, the WEF can be used to compute an upper bound of the word error probability.

In [61], the authors define the Input-Redundancy Weight Enumerating Function (IRWEF) of the code as:

$$A^c(W,Z) = \sum_{w,j} A_{w,j} \; W^w \; Z^j \tag{6-3}$$

where $A_{w,j}$ denotes the number of codewords of Hamming weight j generated by an input information word of Hamming weight w. The IRWEF makes explicit, in each term of the WEF, the separate contributions of the information and the parity bits to the total Hamming weight of the codewords. The IRWEF then provides information on the bit error probability.

The conditional weight enumerating function, $A_w^c(Z)$, of the parity check bits generated by the constituent code C corresponding to the input words of weight w is defined as follows:

$$A_w^c(Z) = \sum_{j} A_j^c \; Z^j \tag{6-4}$$

where A_j^c denotes the number of codewords of weight j, and can be related to the IRWEF by:

$$A^c(W,Z)=\sum_w W^w\ A_w^c(Z) \tag{6-5}$$

Both $A^c(W,Z)$ and $A_w^c(Z)$ can be used with the union bound to compute an upper bound to the bit error probability for maximum likelihood decoding. From this, the probability of bit error of turbo codes, in terms of its conditional weight function can be shown as follows [61, 62]:

$$P_{bit} \le \sum_j \sum_w \frac{w}{N} A_{w,j}^{c_p} P_2(j) \tag{6-6}$$

where $A_{w,j}^{c_p}$ denotes the number of codewords of the parallel concatenation (turbo code) with weight j generated by a word of information weight w. Parameters d_1 and d_2 represent the parity weights of the first and second encoders respectively. The weight $j = d_1 + d_2 + w$ and $P_2(j)$ is the pair-wise error probability between the zero-codeword and the codeword with weight j. This means that to calculate the probability of bit error, P_{bit}, the WEF, $A_w^{c_p}(Z)$, must be known.

For a turbo code with a fixed interleaver, the construction of $A_w^{c_p}(Z)$ requires an exhaustive search. Due to the complexity involved in this search, [61] introduced the concept of the uniform interleaver to reduce the search overhead. Using this approach, an average upper bound on performance could be constructed by averaging over all possible interleavers.

Definition 1: A uniform interleaver of length N is a probabilistic device which maps a given input word of weight w into all distinct N_{cw} permutations of it with equal

probability $1/N_{cw}$. From this, the average conditional WEF of the parallel concatenation (turbo code) can be obtained from those of the constituent codes

$$A_w^{c_p}(Z) = \frac{A_w^{c_1}(Z)\ A_w^{c_2}(Z)}{N_{cw}},\ N_{cw} = \binom{N}{w} \tag{6-7}$$

or equivalently, $$A_w^{c_p}(Z) = \sum_j A_{w,j}^{c_p}\ Z^j \tag{6-8}$$

To get the individual components of Equation 6-6, $A_{w,j}^{c_p}$, equate the right hand sides of Equations 6-7 and 6-8. From this, the WEF of turbo code can be written as:

$$A^{c_p}(W,Z) = \sum_w W^w A_w^{c_p}(Z) \tag{6-9}$$

6.4 Derivation of The Punctured Bound

For punctured convolutional codes, extensive computer searches [104-107] have been performed to get an optimal (maximum free distance) puncturing pattern to obtain higher rates from the original low-rate code. Once the puncturing pattern is obtained, it is applied to the original low-rate trellis to get the WEF of the higher rate code. For turbo codes, the computer search for optimum puncturing patterns is very complex due to the existence of the interleaver. One of the goals of this research is to get the WEF of the punctured turbo code that is independent of specific puncturing pattern.

In [61, 62], the authors calculated the average performance upper bound for turbo codes with rate 1/3. Here, from the knowledge of $A_{w,j}^{c_p}$, the methodology used in [61, 62] to calculate the probability of bit error rate of the punctured turbo codes for rates higher than 1/3 is extended.

The number of codewords of a punctured turbo code with punctured weight v, generated by a word of information weight w, is denoted by $A_{w,v}^{c_{pp}}$. Writing the conditional WEF of the punctured turbo code yields:

$$A_w^{c_{pp}}(Z) = \sum_v A_{w,v}^{c_{pp}} \; Z^v \tag{6-10}$$

where Z is a dummy variable. The individual terms of the average number of codewords are then calculated with punctured weight v generated by information weight w of the punctured turbo code, $A_{w,v}^{cpp}$. Rewriting the probability of bit error in terms of the punctured components yields:

$$P_{bit} \le \sum_v \sum_w \frac{w}{N} A_{w,v}^{cpp} \; P_2(v) \tag{6-11}$$

By knowing the conditional WEF, $A_{w,j}^{cp}(Z)$, of the parallel concatenation of rate 1/3, the conditional WEF of the punctured code, $A_{w,v}^{cpp}(Z)$, can be derived for any given code rate $r > 1/3$. The calculation is accomplished by introducing the concept of a *hypergeometric puncturing device.* Using this first-of-its-kind device, tractable analytic solutions to performance bounds are derived. The *hypergeometric puncturing device* is defined as follows:

Definition 2: For a given codeword of length L and weight j generated from a word of information weight w, the device is to puncture (delete) M bits from the L bits (i.e., randomly choose L-M bits from L bits to survive). The *hypergeometric puncturing device* is a probabilistic device that maps the L bits with weight j to L- M bits of all possible weights v with probability given by:

$$P_w(V = v \mid j) = \frac{\binom{j}{v}\binom{L-j}{L-M-v}}{\binom{L}{L-M}}, \qquad 0 \le v \le j \tag{6-12}$$

The hypergeometric distribution [108] is a well known distribution in probability theory. Given a set with N items, with K being defective (or different), n items are randomly chosen from the set. The hypergeometric distribution yields the probability of getting x of the n items from the K set (or defective). Thus, X is a hypergeometric random variable and its probability distribution is given by Equation 6-13:

$$P(X = x \mid K) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}, \qquad x = 0,1,\ldots,n \tag{6-13}$$

Example: Suppose there exist an input codeword of length $L = 8$, and weight $K = 3$, with an original code rate of 1/3. Let this codeword enter the puncturing device to yield a rate 1/2 codeword. Then, as a representative example, $M = 4$ bits from this codeword need to be punctured. The possible output weights after puncturing are $x = 0, 1, 2,$ and 3. Substituting these values into Equation 6-13,

$$P(X=x \mid K=3)=\frac{\binom{3}{x}\binom{5}{4-x}}{\binom{8}{4}}$$

Results in

$P(X=0 \mid K=3)=5/70,$

$P(X=1 \mid K=3)=30/70,$

$P(X=2 \mid K=3)=30/70,$

$P(X=3 \mid K=3)=5/70,$

And $\sum_{x} P(X=x \mid K=3)=1$

From Definition 2, the output of the puncturing device is averaged over all possible punctured positions to yield all the possible output punctured weights. As a consequence of this, the components of the conditional WEF of the punctured turbo codes are obtained from:

$$A_{w,v}^{cpp}=\sum_{j} A_{w,j}^{c_p} P_w(V=v \mid j) \tag{6-14}$$

Similar to the uniform interleaver in [61], one is confronted with a deterministic puncturing map. This map gives rise to one particular punctured frame for the input frame (i.e., only one punctured weight of the punctured frame). But the introduction of the hypergeometric puncturing device facilitates the derivation of the punctured WEF of the parallel concatenation. So, the following theorem and proof illustrates the

contribution of this device to averaging the number of punctured codewords with punctured weight v generated from a codewords with information weights w.

Theorem: Letting $A^{C_{ppk}}(W,V)$ be the IRWEF of the punctured code C_{p_k} obtained using the particular puncturing pattern P_k, then:

$$E_k\left[A^{C_{ppk}}(W,V)\right] = A^{C_{pp}}(W,V) \tag{6-15}$$

Where E_k[*] is the mean with respect to all punctured positions that gives all the possible output weights.

Proof:

$$E_k\left[A^{C_{ppk}}(W,V)\right] = E_k\left[\sum_w W^w A_w^{C_{ppk}}(V)\right] \tag{6-16}$$

$$= \sum_w W^w E_k\left[A_w^{C_{ppk}}\right] \tag{6-17}$$

$$= \sum_w W^w E_k\left[\sum_v A_{w,v}^{C_{ppk}}\right] \tag{6-18}$$

$$= \sum_w W^w \sum_v E_k\left[A_{w,v}^{C_{ppk}}\right] \tag{6-19}$$

From the definition of the *hypergeometric puncturing device,* $A_{w,v}^{C_{pp}} = E_k\left[A_{w,v}^{C_{ppk}}\right]$, then

$$E_k\left[A^{C_{ppk}}(W,V)\right] = \sum_w W^w \sum_v A_{w,v}^{C_{pp}} \tag{6-20}$$

$$= \sum_w W^w A_w^{C_{pp}}(V) = A^{C_{pp}}(W,V) \tag{6-21}$$

Since the analytical upper bound has a linear dependency with the conditional WEF of the punctured code as shown in Equation 6-11, the following corollary is achieved:

Corollary: The analytical upper bound using the IRWEF of the punctured turbo code, $A^{C_{pp}}(W,V)$ coincides with average of the analytical upper bounds calculated with hypergeometric puncturing device.

6.5 Application of The Bound

The derived punctured bound in Equation 6-11 can be studied on various statistical channels by formulating the pair-wise error probability, $P_2(v)$, for the channel of interest. This section applies the derived punctured bound, using Binary Phase Shift Key (BPSK) modulation, to the Additive White Gaussian Noise (AWGN) channel, the fully interleaved Rayleigh fading channel, and the correlated Rayleigh fading channel.

6.5.1 Additive White Gaussian Noise (AWGN) Channel

It is assumed that the channel has AWGN with two-sided noise power spectral density of $N_0/2$. Using BPSK modulation, the pair-wise probability is given by [12]:

$$P_2(v) = Q\left(\sqrt{\frac{2vrE_b}{N_0}}\right) \tag{6-22}$$

where r is the code rate of the code, E_b/N_0 is the signal-to-noise ratio per information bit, v is the codeword weight, and $Q(x)$ is the tail integral of a standard Gaussian density with zero mean and unit variance defined as:

$$Q(x)=\int_{x}^{\infty}\frac{1}{\sqrt{2\pi}}\,e^{-z^2/2}\,dz \tag{6-23}$$

6.5.2 Fully Interleaved Fading Channel with Perfect Side Information

For the fully interleaved channel with perfect side information, the exact probability of incorrectly decoding a codeword C_0 (zero codeword) into a codeword C_j which differs from C_0 in v bit positions indexed by $(i_1, i_2, \cdots, iv)$ is given by [75]:

$$P(C_0 \to C_j|a)=Q\left(\sqrt{\frac{2rE_b}{N_0}\sum_{k=1}^{v}a_{i_k}^2}\right) \tag{6-24}$$

where a_{i_k} is the fading amplitude. To compute the average word error probability, $P(C_0 \to C_j)$ must be averaged over the fading amplitude a. The result is a multi-dimensional integral and given as:

$$P(C_0 \to C_j)=\int_{a_{i_1}}\cdots\int_{a_{i_v}} p_A(a_{i_1}, a_{i_2}, \cdots, a_{i_v})\,.\,P(C_0 \to C_j|a)\,da_{i_1}\cdots da_{i_v} \tag{6-25}$$

where $p_A(a_{i_1}, a_{i_2}, \cdots, a_{i_v})$ is the joint probability density function of the fading amplitudes. In case of independent fading amplitudes, the indices of the different bit positions are of no importance (only the weight of the incorrect codeword matters). The pair-wise error

probability can be formulated in terms of the Hamming distance of the codewords as follows:

$$P_2(v) = \int_{a_1} \cdots \int_{a_v} p_A(a_1, a_2, ..., a_v) \cdot Q\left(\sqrt{\frac{2rE_b}{N_0}} \sum_{k=1}^{v} a_k^2\right) da_1 \cdots da_v \qquad (6\text{-}26)$$

For independent fading amplitudes,

$$p_A(a_1, a_2, \cdots, a_v) = \prod_{i=1}^{v} p_A(a_i) \qquad (6\text{-}27)$$

The fading amplitude a_i is modeled with a Rayleigh probability density function, where $p_A(a_i) = 2a_i\ e^{-a_i^2}$ for $a_i > 0$. With sufficient interleaving (fully interleaved), the a_i's are independent. The exact calculation of Equation 6-26 is extremely difficult. To solve this problem, the authors [75] examine four options. The first option (exact) is to simplify Equation 6-26 to a form that can be evaluated via numerical integration. The other three options avoid the problem of numerical integration by seeking closed form upper bounds for $P_2(v)$. The authors reported three other simplified options for the pair-wise error probability and also concluded with comparison evaluations of these different simplification versions:

Option 1: This bound is tight for larger values of rE_b/N_0 and weak at lower values and is given by:

$$P_2(v) \le C(v) \left(\frac{rE_b}{N_0}\right)^{-v} \qquad (6\text{-}28)$$

where $$C(v)=\frac{1}{2}\left(\frac{1\cdot 3\cdots(2v-1)}{2\cdot 4\cdots(2v)}\right) \tag{6-29}$$

Option 2: This bound differs by a scale factor from the exact for all values of rE_b/N_0 and is given by:

$$P_2(v)\le\frac{1}{2}\left[\frac{1}{1+\frac{rE_b}{N_0}}\right]^v \tag{6-30}$$

Option 3: This bound is tight for lower values of rE_b/N_0 and differ by a scale factor from the exact for higher values of rE_b/N_0 and is given by:

$$P_2(v)\le\frac{1}{2}\left[1-\left(\frac{\frac{rE_b}{N_0}}{1+\frac{rE_b}{N_0}}\right)^{1/2}\right]\left[\frac{1}{1+\frac{rE_b}{N_0}}\right]^{v-1} \tag{6-31}$$

6.5.3 Correlated fading channel with perfect side information

The exponentially correlated fading channel with an autocorrelation function is given by $\rho(\tau)=e^{-|2\pi f_d\tau|}$, where f_d is the Doppler bandwidth and τ is the lag parameter. As shown in [75, 109, 110], the pair-wise error probability is bounded by an expression which is a function of v and f_dT_s, where T_s is the symbol duration, and given by:

$$P_2(v)\le\frac{1}{2}\left[\left(\frac{rE_b}{N_0}\right)^v\left(1-e^{-|4\pi f_dT_s|}\right)^{v-1}+v\left(\frac{rE_b}{N_0}\right)+1\right]^{-1} \tag{6-32}$$

6.6 Performance Evaluation

This section shows the results of the derived punctured bound for different turbo code rates derived from the original code rate 1/3 turbo code. Using the simulation model, obtained results are compared with derived analytic bounds. The simulation model uses a turbo code with two identical parallel concatenated recursive systematic convolutional constitute encoders, separated by a random interleaver. Each encoder uses octal generators 5 (feedforward) and 7 (feedback) and is denoted as $(5/7)_8$. Both encoders are terminated using the scheme presented in [38]. The encoded bits are punctured and then modulated using a BPSK modulator, and then transmitted over the designated channel. The decoding is performed using a Soft Output Viterbi Alogorithm (SOVA) as a constitute decoder [51] with 8 iterations.

The derived punctured bound of the turbo code with two identical constituent encoders is applied with generator functions $(5/7)_8$. The algorithm in [62] is used to calculate the WEF of the constituent codes.

For an AWGN channel, Figures 6-2, 6-3 and 6-4 show the bound for frame lengths of 100, 200, and 500 bits for different code rates of 1/3, 2/5, 1/2, and 2/3 respectively. As expected, the punctured bounds diverge at signal-to-noise ratios larger than those that occur at rate 1/3.

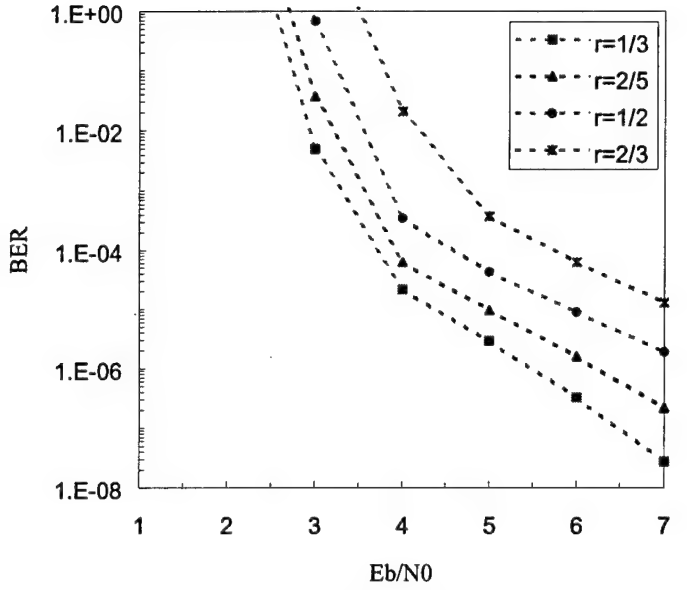

Figure 6-2 Analytical bounds for frame length of 100 bits in AWGN channel.

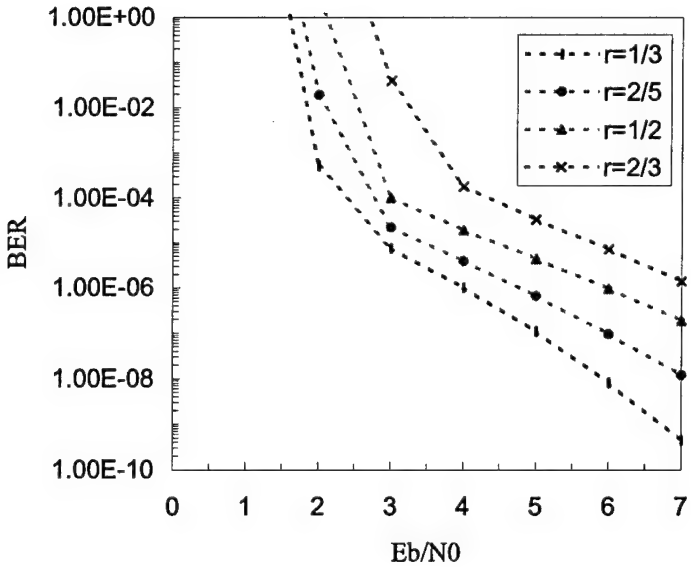

Figure 6-3 Analytical bounds for frame length of 200 bits in AWGN channel.

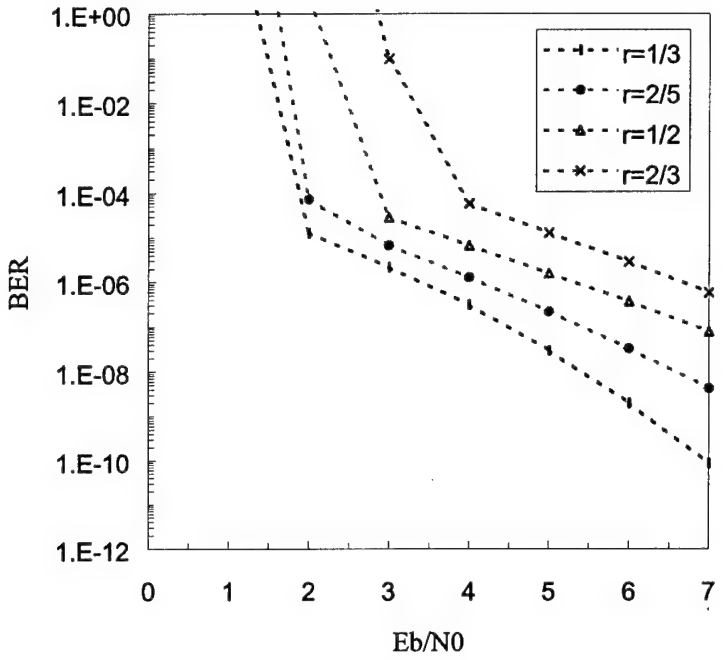

Figure 6-4 Analytical bounds for frame length of 500 bits in AWGN channel.

The abrupt transition of the bound occurs when the signal-to-noise ratio, E_b/N_0, drops below the threshold determined by the computation cutoff rate R_0, i.e., when $E_b/N_0 < -1/r \ln (2^{1-r} - 1)$ for a code rate r [62]. Figures 6-2, 6-3, and 6-4 show the abrupt change occurring for rates 1/3, 2/5, 1/2, and 2/3 at 2.03 db, 2.2 db, 2.5 db, and 3.1 db, respectively. Also, as seen for the 1/3 rate code, the evaluation of the punctured bound requires only a few terms in the summation at higher signal-to-noise ratios. The error floor (the low slope region of the performance curve where the error rate decreases very slowly with increasing the signal-to-noise ratio) still exists with the punctured bound.

Figure 6-5 shows a simulated punctured turbo code with a 192-bit frame length compared with the analytical punctured bound at rates 2/5 and 1/2. At higher signal-to-noise ratios (greater than approximately 2 dB), the bound accurately predicts the turbo decoder performance. At signal-to-noise ratios less than R_0, simulation is the only way to predict the performance of turbo codes due to the divergence in the performance of the analytical bound in this region.

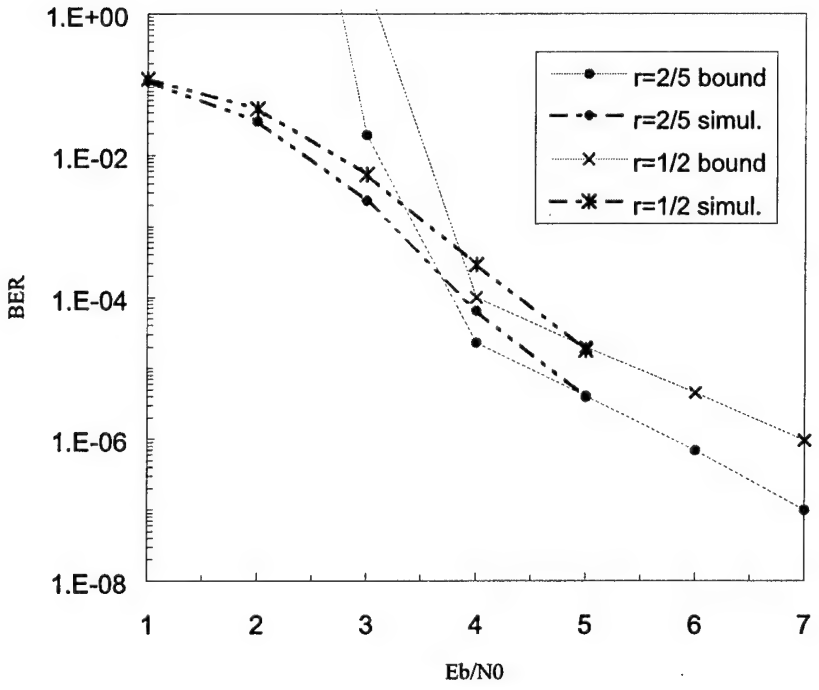

Figure 6-5 Simulated and analytical bound for frame length of 192 bits punctured turbo code in AWGN channel.

For the fully-interleaved fading channel, Figures 6-6, 6-7, and 6-8 show the analytical bound for frame lengths of 100, 200, and 500 bits at code rates of 1/3, 2/5, 1/2, and 2/3. To calculate these bounds, the third option (Equation 6-31) is used as recommended in [75]. This approach yields the best compromise between the performance in regions of low signal-to-noise ratios while not requiring numerical integration. Figures 6-6, 6-7, and 6-8 show the effects of puncturing and frame lengths. As the code rate increases (more deleted bits), the performance degrades and as the frame lengths increase, the performance improves. It is also shown that the bounds diverge at lower signal-to-noise ratios. This leads to the need for the simulation to study the performance of punctured turbo codes in the diverged regions.

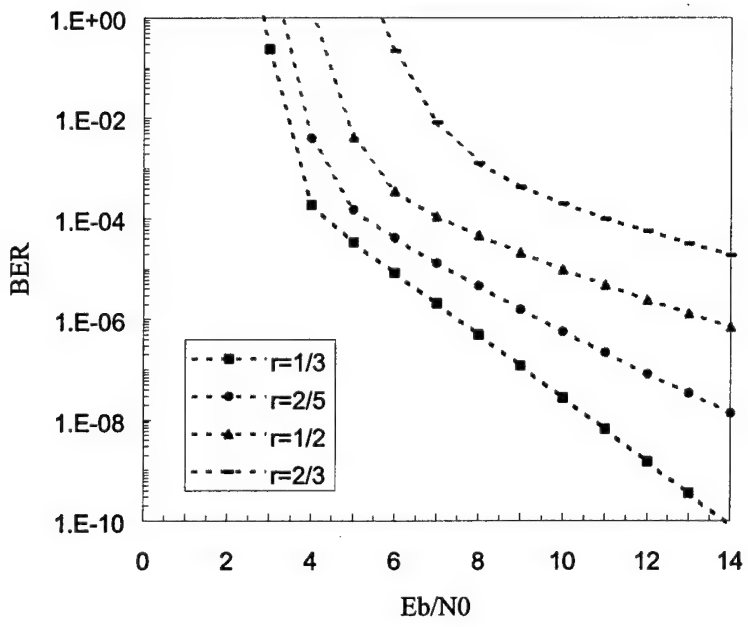

Figure 6-6 Analytic bounds for frame length of 100 bits in the fully interleaved fading channel.

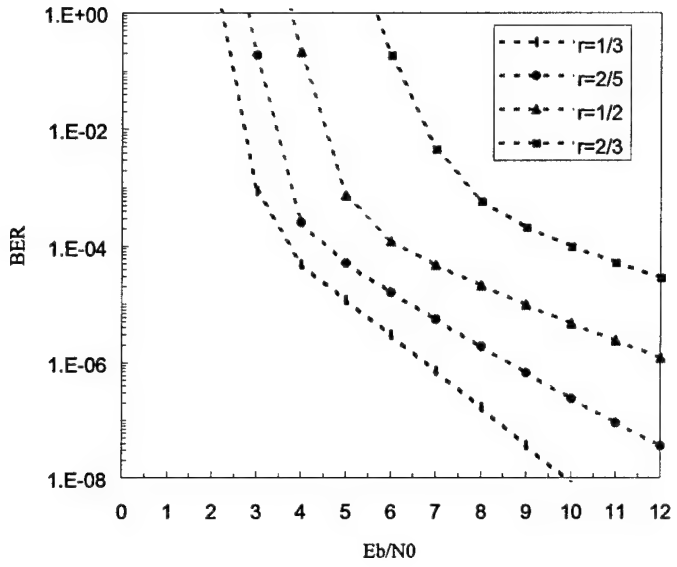

Figure 6-7 Analytic bounds for frame length of 200 bits in the fully interleaved fading channel.

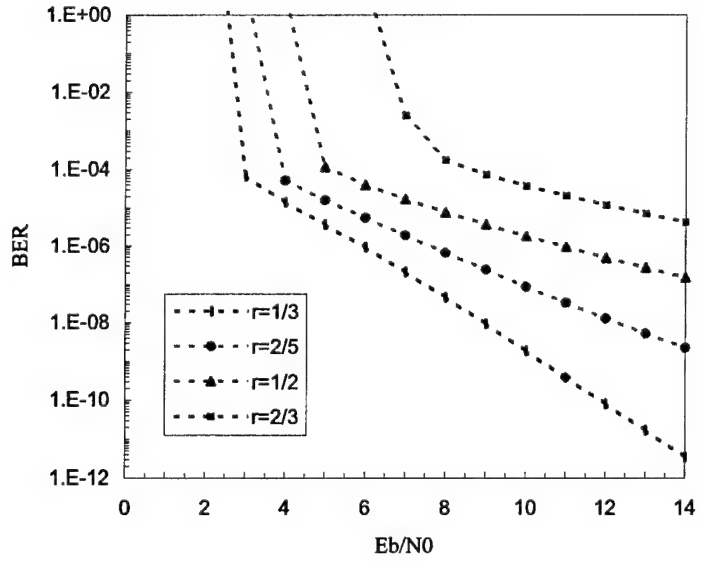

Figure 6-8 Analytic bounds for frame length of 500 bits in the fully interleaved fading channel.

In Figure 6-9, a simulated punctured turbo code with a frame length of 192 bits is compared with the analytical punctured bound at code rates of 1/2 and 2/3. Figure 6-9 shows that the analytic bound diverges at lower signal-to-noise ratios and the simulation is the only tool to investigate the performance at lower signal-to-noise ratios.

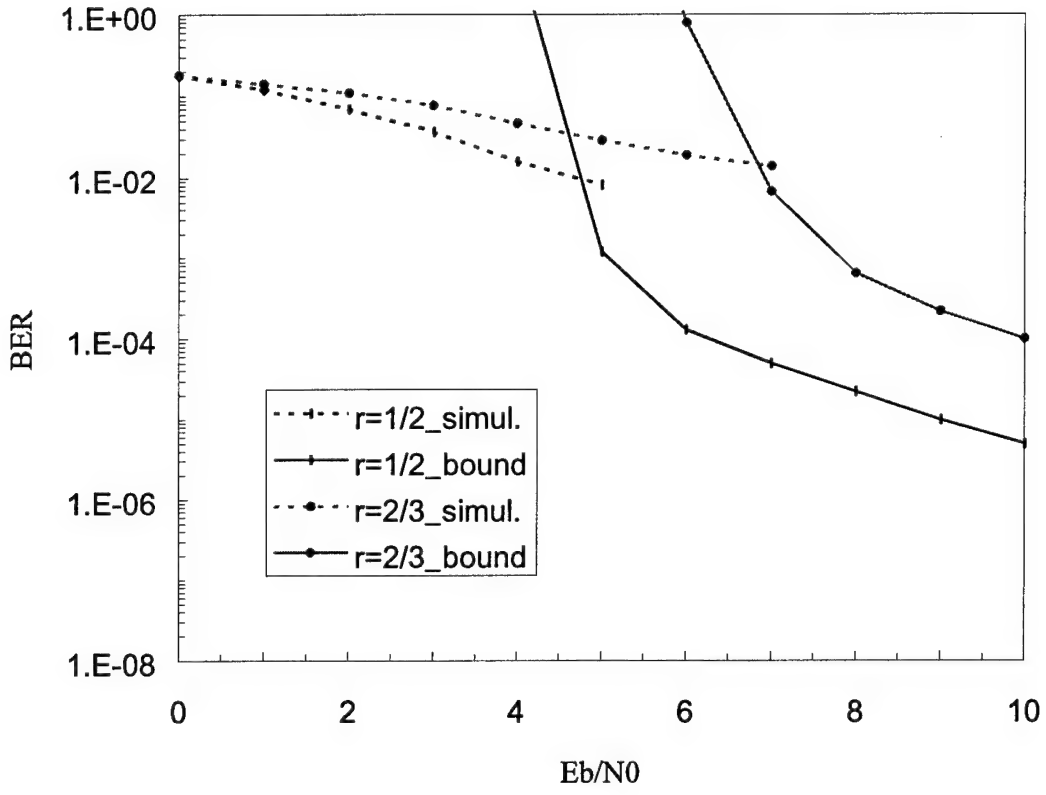

Figure 6-9 Simulated and analytical bound for frame length of 192 bits punctured turbo code in the fully interleaved fading channel.

For the correlated fading channel, Figures 6-10, 6-11, and 6-12 show the analytical bound for frame lengths of 100, 200, and 500 bits, respectively, at different code rates of 1/3, 2/5, 1/2, and 2/3 for various fading rates (f_dT_s products) of 0.1 and 0.01. It can be seen that the performance degrades as the code rate increases (more deleted bits), the frame length decreases, and the correlation of the channel increases (f_dT_s decreases).

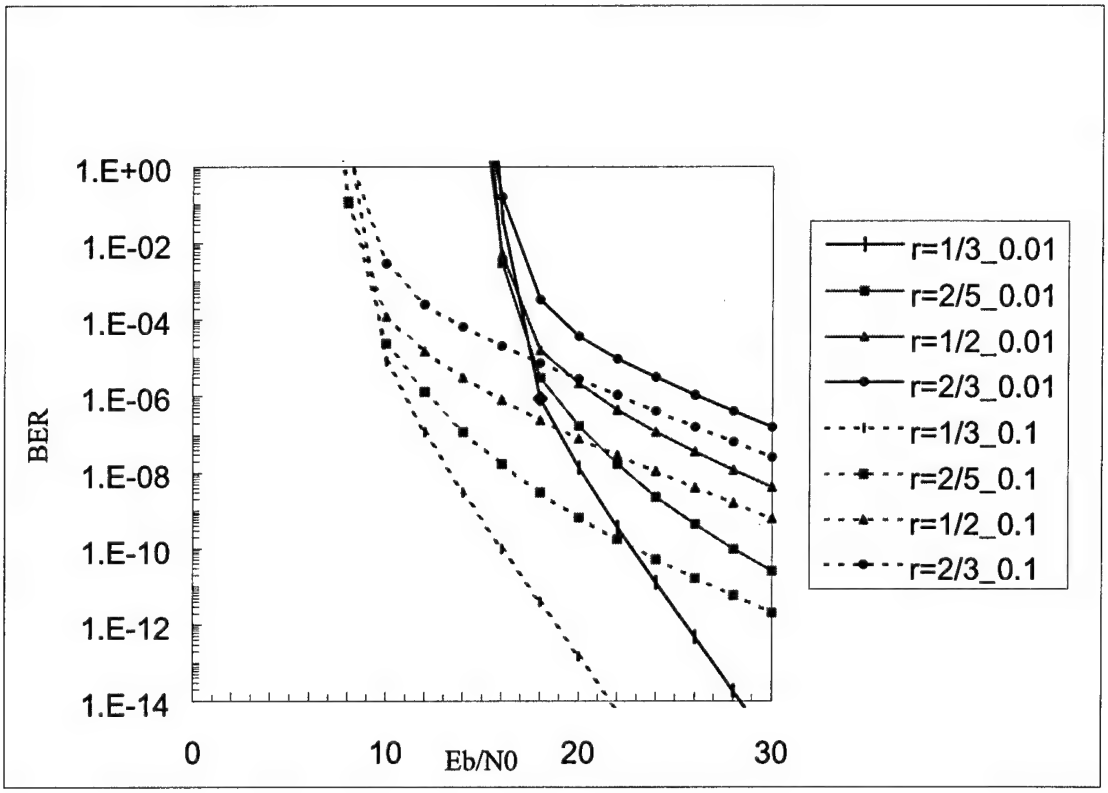

Figure 6-10 Analytical bounds for frame length of 100 bits in the correlated fading channel with 0.01 and 0.1 correlation rates.

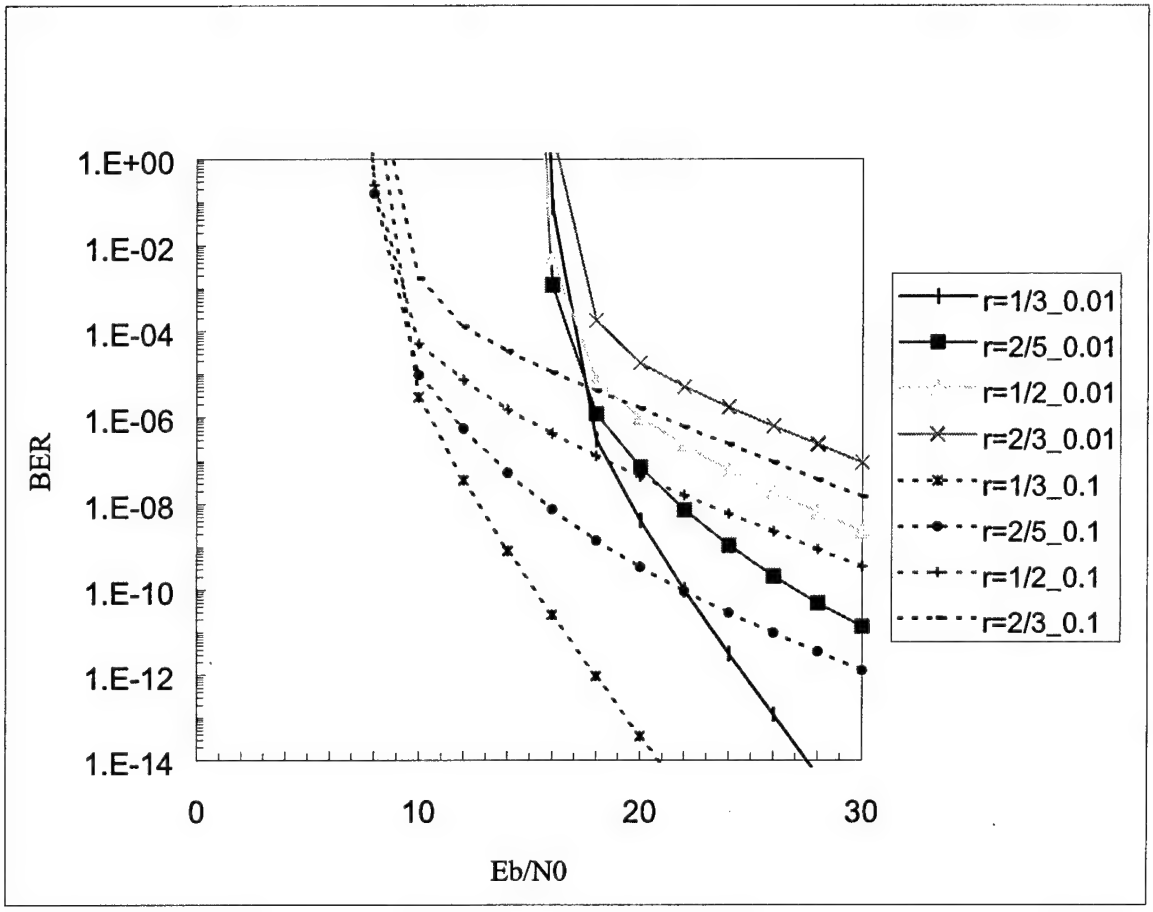

Figure 6-11 Analytical bounds for frame length of 200 bits in the correlated fading channel with 0.01 and 0.1 correlation rates.

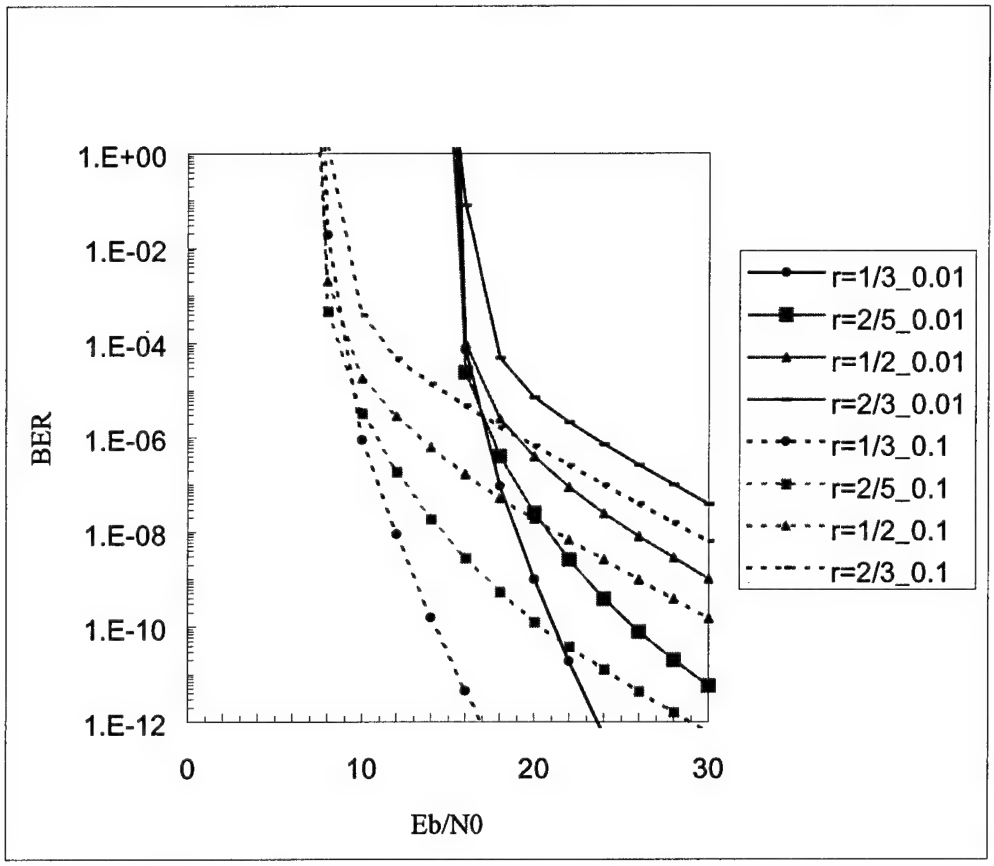

Figure 6-12 Analytical bounds for frame length of 500 bits in the correlated fading channel with 0.01 and 0.1 correlation rates.

Figures 6-13 and 6-14 show the simulated punctured turbo codes with a frame length of 192 bits for fading rates of $f_d T_s = 0.01$ and 0.1, respectively. In the simulated model, the auto-correlation function, $R(k)$, given in Equation 6-33 is used. This function is based on Clark's model [94] and is given by:

$$R(k) = \frac{1}{\pi f_d T_s} J_0(2\pi f_d T_s k) \tag{6-33}$$

where $J_0(*)$ is the Zero-order Bessel function of the first kind. This model is used in the simulation to produce more realistic results than those obtained from the exponential correlation. While exponential correlation has the benefit of mathematical tractability, it is not the most realistic model [111].

Figures 6-15 and 6-16 present a comparison of a simulated punctured turbo code with a frame length of 192 bits to the analytical punctured bound at rates of 2/5 and 1/2 for $f_d T_s = 0.01$ and 0.1, respectively. Figures 6-15 and 6-16 show that as the correlation of the channel increases ($f_d T_s$ decreases), and the code rate increases (due to more punctured bits), the need for simulation to study the performance at lower signal-to-noise ratios increases due to the fast divergence of the analytical bound.

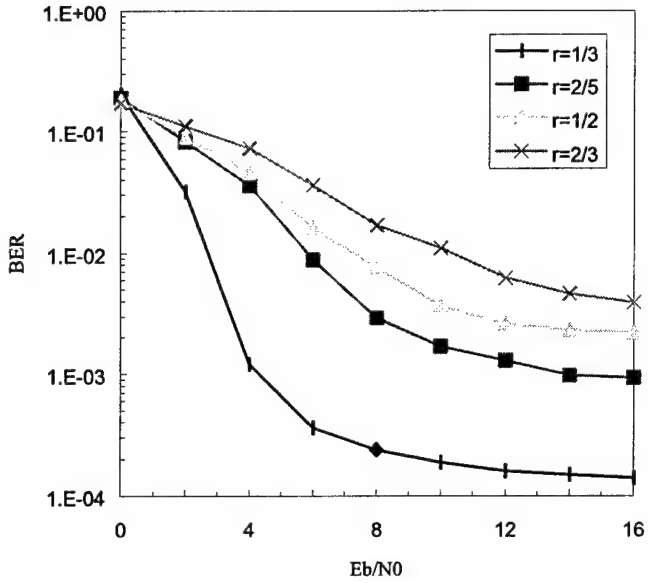

Figure 6-13 Simulated bounds of punctured turbo code for frame length of 192 bits in the correlated fading channel with 0.01 correlation rate.

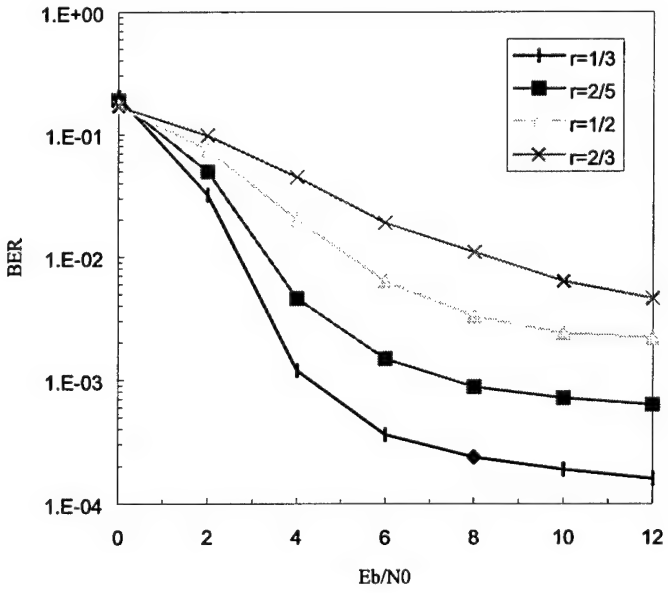

Figure 6-14 Simulated bounds of punctured turbo code for frame length of 192 bits in the correlated fading channel with 0.1 correlation rate.

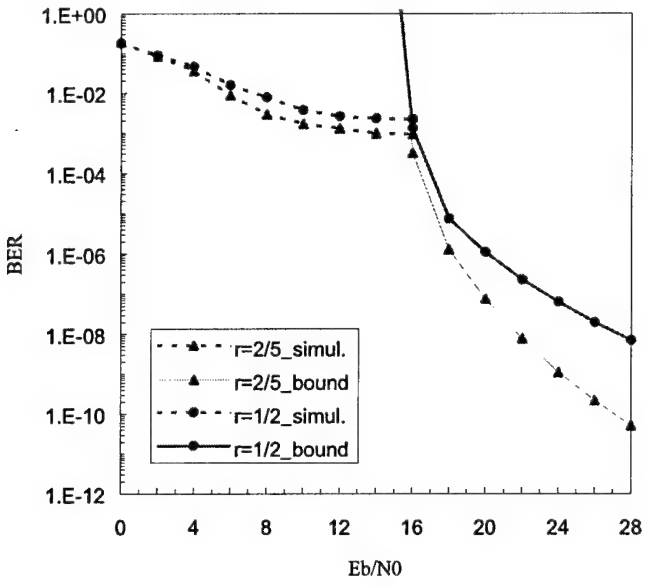

Figure 6-15 Simulated and analytical bound for frame length of 192 bits in the correlated fading channel with $f_d T_s = 0.01$.

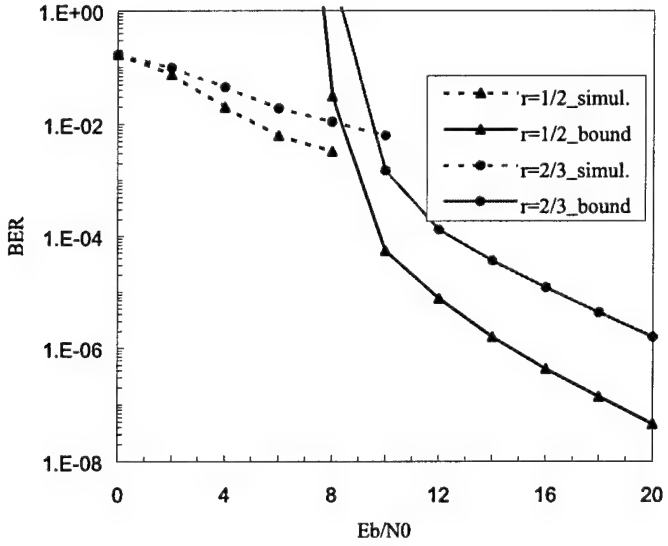

Figure 6-16 Simulated and analytical bound for frame length of 192 bits in the correlated fading channel with $f_d T_s = 0.1$.

6.7 Summary

In this chapter, the hypergeometric puncturing device was introduced. The introduction of the hypergeometric puncturing device makes the derivation of the analytical punctured bound of turbo codes tractable. The hypergeometric puncturing device allows for averaging over all possible punctured positions. Simulation results in AWGN, fully-interleaved fading, and correlated fading channels were also presented along with the analytical bound. The analytic performance bound was compared with the simulation results obtained for various code rates. The comparison shows that the two bounds, analytical and simulation, are identical at higher signal-to-noise ratios but diverge at lower signal-to-noise ratios. This bound can also be extended for use with different punctured turbo code modulations and for assistance in designing punctured turbo codes.

7 Conclusions and Future Work

7.1 Conclusions

In Chapter 2, the parameters and types of fading in wireless communication systems were reviewed. The channel coding techniques deployed in wireless communication systems were also discussed.

Turbo code performance is sensitive to its code structure which is made up of code rate, constraint length, tap connection, block size, interleaving pattern, and number of decoding iterations. In Chapter 2, the problem of choosing different turbo code components and how these choices affect turbo code performance was discussed. The problem of turbo code application to cellular mobile communication systems was also discussed.

From this discussion, it was concluded that optimizing different turbo code components is closely related to the chosen application. The choice of constituent encoder depends on its distance spectrum. If the quality of service requires low bit error rate, e.g., data transmission, the design focuses on choosing a constituent encoder with a sparse spectrum for the first few low-weight codewords. The termination of both encoders was also shown to be important. In the case where lower signal-to-noise ratios

or higher bit error rates are permissible, e.g., speech transmission, the distance spectrum of the constituent encoders plays a very important role in choosing this component. In this region, the multiplicity of low and moderate weight codewords is required to be minimal. Also, choosing the interleaver length is restricted by application sensitivity to the time delay and restricts performance in certain cases, e.g., speech transmission. Interleaving map performance depends on interleaver length. For short frames, structured interleavers perform better than nonstructured ones. Random interleavers perform best when the interleaver is long.

It was shown that sensitivity to the delay restricts the number of decoder iterations. Complexity and memory size availability also restrict the component decoder type used.

The principle of iterative decoding using turbo decoder was presented in Chapter 3. Turbo decoder performance depends mainly on the soft-input/soft-output constituent decoders. The process for applying the soft-in/soft-out constituent decoder to the iterative turbo decoder was presented. Information transfer from one decoder to another was shown to be essential for improving performance from one step to the next in the iterative decoding process. The Viterbi algorithm was also reviewed, along with its modified version that delivers soft decisions used by a constituent soft-input/soft-output decoder. All related mathematics associated with implementing the iterative decoding process were presented.

Chapter 4 provided a discussion on the analytical model development for evaluating the analytical bound for turbo code performance. The idea of uniform

interleaving was presented along with tractable analytical bound calculations. A method was detailed for calculating the average weight enumerating function of turbo codes using the weight enumerating function calculation for convolutional codes when used as constituent decoders of the parallel concatenation.

The development of turbo code simulation models, channel models, and implementation issues were presented. The Soft Output Viterbi Algorithm (SOVA) implementation in a constituent decoder of the turbo decoder was addressed along with practical implementation issues. The last section of Chapter 4 detailed operating assumptions for experiments conducted.

In Chapter 5, the problem of energy allocation in turbo codes was examined. In standard turbo coding, all bits are transmitted with equal energy. This strategy does not guarantee optimum allocation. For turbo code applications involving short frames, computer simulation and analytic modeling were used to examine different ways to allocate energy to frame bits. Results show that, for turbo codes with short frames (using SOVA as a constituent decoder component) operating in very low signal-to-noise environments, allocating more (without restriction on the amount of increase) energy to the systematic bits improves performance. At higher signal-to-noise ratios, allocating less energy to systematic bits improves performance. The most important results are seen at a BER of approximately 10^{-3}. In this region, results indicate that by allocating half the energy to systematic bits, optimal performance can be achieved. Any deviation from this half energy point results in degraded performance.

A new interleaving approach, the general circular shift interleaver, is presented in Chapter 5. This interleaver's performance was examined and reported using mathematical analysis and computer simulations. The application of the circular shift interleaver and its generalized version for both equal and unequal error protections with turbo codes for short frame lengths were presented. In the case of equal error protection, the results show that circular shift interleaving provides slightly better performance at higher E_b/N_0 values when compared with block and random interleaving. In the case of unequal error protection, the generalized circular shift interleaver offers comparable performance to those using a different inteleaver for each level of protection.

Chapter 6 presented a novel approach for obtaining the analytic punctured bound for turbo codes. The hypergeometric puncturing device was introduced which makes the analytical punctured bound derivation of turbo codes tractable. The hypergeometric puncturing device allows for averaging over all possible punctured positions. Simulation results in AWGN, fully interleaved fading, and correlated fading channels are also presented along with the analytical bound. The analytic performance bound was compared with simulation results for various code rates. The comparison shows that the two bounds, analytical and simulation, are identical at higher signal-to-noise ratios with the analytical bound diverging at lower signal-to-noise ratios. The abrupt transition of the analytic bound occurs when the signal-to-noise ratio drops below the threshold determined by computation cutoff rate. Simulation is the only effective way to predict turbo code performance in regions where the analytic bound divergence.

7.2 Contributions and Future Work

The original contributions of this research:

1. A comprehensive turbo code tutorial with application to wireless mobile communication systems. A tutorial of this level is not available in previous open literature. As a result, a draft journal article is under revision for *IEEE Communications Surveys*.

2. A complete tutorial of the turbo decoder iterative processing. This discussion addressed the most complex and least understood concepts of turbo coding.

3. Turbo code performance enhancement using short frames with unequal energy distribution. Simulation and analysis results have been published in *IEEE International Conference on Information Technology: Coding and Computing* [112].

4. Proposal of a new interleaver for short frame applications. This new interleaver can be used in both equal and unequal error protections application. This work has been published in *IEEE International Conference on Information Technology: Coding and Computing* [113].

5. A novel method for deriving an analytic bound of punctured turbo codes has been introduced. This work has been published in the journal of *IEE Electronic Letters* [114].

6. A mathematical analysis for punctured turbo coding which fills a critical void in previously published works. This analysis is summarized in a draft paper to be submitted to *IEEE Transactions on Information Theory.*

Proposals for future work may include:

1. For short frame applications, structured interleavers can perform as well as random interleavers. This opens areas for future work to search or design an optimum structured interleaver. In the case of punctured turbo codes, the existence of the interleaver complicates the puncturing process. Thus, any search or design procedure for an optimum interleaver must take into account the puncturing process.

2. Study the performance of turbo codes with short frames using a jointly designed constituent encoder and interleaver. Different constituent encoders, different polynomial generators and different memory sizes, should be studied along with any proposed interleaver. Throughout this investigation, the constituent encoder used a memory length of two and its generator polynomial was optimum in the sense that it maximized the free distance and effective weight of the constituent code.

3. Extending the derived punctured bound for use with different punctured turbo code modulations and for aiding in the design of punctured turbo code components (constituent encoders).

4. Extending the iterative decoding process of turbo codes, especially with speech transmission (i.e., short frames applications), to be used jointly with the source decoder. This joint decoding should enhance turbo code performance. In a typical case, both the channel decoder and source decoder work independently. Exchanging information between the source decoder and the channel decoder in an iterative process before making a final decision will enhance performance. As A. J. Viterbi said in [115], one of three lessons learned was to never discard information prematurely that may be useful in making a decision until after all decisions related to that information have been computed.

Appendix A

The Maximum A posteriori Probability (MAP) Algorithm

Throughout this investigation, the Soft Output Viterbi Algorithm (SOVA) was mentioned many times versus the Maximum A posteriori Probability (MAP) algorithm. The only algorithm discussed previously was the SOVA. Here the MAP algorithm is presented for completeness.

The Viterbi algorithm (VA) is an optimal decoding algorithm which minimizes the probability of sequence error for convolutional codes. However, this algorithm does not necessarily minimize the probability of symbol (bit) error for the decoded bit. Also, VA is not able to deliver the *a posteriori* probability (APP) for each decoded bit.

The symbol-by-symbol maximum *a posteriori* (MAP) algorithm was proposed in 1974 by Bahel, *et al.* [47] as an optimal decoding algorithm. This algorithm is also denoted by BCJR (first letters of the name of the four authors). The algorithm minimizes the symbol error probability in decoding linear block and convolutional codes and delivers the APP for each decoded bit. In [2, 27], the BCJR or MAP algorithm was modified in order to take into account the recursive character of the convolutional code to be applied to turbo codes.

MAP algorithm for recursive systematic convolutional codes:

Consider a recursive systematic convolutional (RSC) code with constraint length K, and rate 1/2, at time k the encoder state denoted by S_k. Assume that the information bit sequence $\{d_k\}$ has N-bit length, where $d_k = \{0, 1\}$, and the initial state of the encoder, S_0, and the final state of the encoder, S_N, are the zero states.

The input to the encoder at time k is a bit d_k and the corresponding output binary couple is (X_k, Y_k). For a discrete memoryless Gaussian channel and a binary modulation, the decoder input is made up of a couple R_k of two random variables at time k, given as:

$$x_k = (2X_k - 1) + i_k \tag{A-1a}$$

$$y_k = (2Y_k - 1) + q_k \tag{A-1b}$$

where i_k and q_k are two independent noises with the same variance σ^2. The encoder output codeword sequence, noted by $C_1^N = \{C_1, ..., C_k, ..., C_N\}$ where $C_k = (X_k, Y_k)$ is the input to a discrete Gaussian memoryless channel whose output is the sequence $R_1^N = \{R_1, ..., R_k, ..., R_N\}$ where $R_k = (x_k, y_k)$.

Now, the task of the MAP algorithm is to calculate the *a posteriori* probability (APP) of each decoded bit. The logarithmic of likelihood ratio (LLR), $\Lambda(d_k)$, associated with each decoded bit d_k is given by:

$$\Lambda(d_k) = \log \frac{P(d_k = 1 | observation)}{P(d_k = 0 | observation)} \tag{A-2}$$

where $P(d_k = i \mid observation)$ and $i = 0, 1$ is the *a posteriori* probability (APP) of the decoded bit d_k. According to the received sequence, R_1^N, Equation A-2 becomes:

$$\Lambda(d_k) = \log \frac{P(d_k = 1 | R_1^N)}{P(d_k = 0 | R_1^N)} \tag{A-3}$$

Now, we need to calculate the APPs $P(d_k = i | R_1^N)$, $i = 0, 1$ for any decoded bit d_k. The encoder is assumed to be a discrete-time finite-state Markov process. The encoder has 2^{ν} distinct states, where ν is the memory of the encoder, $\nu = K - 1$. The object of the decoder is to examine the received codeword, R_1^N, and estimate the APPs of the decoded bits.

At any state m, where $m = \{0, 1, \ldots, 2^{\nu}-1\}$, at time k, the probability of being at state m given the received sequence, R_1^N, is $P(S_k = m | R_1^N)$ which is made up of two components $P(d_k = 1, S_k = m | R_1^N)$ and $P(d_k = 0, S_k = m | R_1^N)$. Then, the APP of a decoded bit at state m at time k is:

$$\lambda_k^i(m) = P(d_k = i, S_k = m | R_1^N), \quad i = 0,1 \tag{A-4}$$

The APP of a decoded bit at time k is the summation of APPs at all states at time k, given as:

$$P(d_k = i \mid R_1^N) = \sum_m \lambda_k^i(m), \quad i = 0,1 \tag{A-5}$$

Substituting from Equation A-5 into Equation A-3, the LLR $\Lambda(d_k)$ associated with each decoded bit d_k becomes:

$$\Lambda(d_k) = \log \frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)} \tag{A-6}$$

Once the APPs $\lambda_k^i(m)$, i =0, 1, are calculated, the decoder can make a decision by comparing $\Lambda(d_k)$ to a threshold equal to zero, as follows:

$$\hat{d}_k = 1 \qquad if \qquad \Lambda(d_k) \geq 0 \tag{A-7a}$$

$$\hat{d}_k = 0 \qquad if \qquad \Lambda(d_k) < 0 \tag{A-7a}$$

To calculate $\lambda_k^i(m)$, i =0, 1, further mathematical manipulation is needed. State m at time k can be reached from any state m' at time k-1 (if a transition from state m' to m is possible in the state diagram of the encoder) as a response to an input $d_k = i$, i =0, 1.

$$\lambda_k^i(m) = P(d_k = i, S_k = m \left| R_1^N \right.)$$

$$= \sum_{m'} P(d_k = i, S_{k-1} = m', S_k = m \left| R_1^N \right.) \tag{A-8}$$

where $P(d_k = i, S_{k-1} = m', S_k = m \left| R_1^N \right.)$ is the APP of the branch connected the two states, m' to m, in the trellis diagram of this encoder. Applying Bayes rule to Equation A-8, then Equation A-8 becomes:

$$\lambda_k^i(m) = \sum_{m'} \frac{P(d_k = i, S_{k-1} = m', S_k = m, R_1^N)}{P(R_1^N)} \tag{A-9}$$

In Equation A-9, the term $P(R_1^N)$ is neglected in the denominator because it is independent of state m or m' and is cancelled when substituting in Equation A-6. The R_1^N in the numerator can expressed as:

$$R_1^N = \{R_1,...,R_k,...,R_N\}$$

$$= \{R_1^k, R_{k+1}^N\} \tag{A-10}$$

where $R_1^k = \{R_1,...,R_k\}$ and $R_{k+1}^N = \{R_{k+1},...,R_N\}$, then Equation A-9 becomes:

$$\lambda_k^i(m) = \sum_{m'} P(d_k = i, S_{k-1} = m', S_k = m, R_1^k, R_{k+1}^N)$$

$$= \sum_{m'} P(R_{k+1}^N | d_k = i,, S_{k-1} = m', S_k = m, R_1^k)\ P(d_k = i, S_{k-1} = m', S_k = m, R_1^k) \tag{A-11}$$

Note that if state S_k is known, then the events after time k are not influenced by observation R_1^k and bit d_k.

Equation A-11 then becomes:

$$\lambda_k^i(m) = \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(d_k = i, S_{k-1} = m', S_k = m, R_1^k) \tag{A-12}$$

Again, R_1^k can be expressed as:

$$R_1^k = \{R_1,...,R_k\}$$

$$= \{R_1^{k-1}, R_k\} \tag{A-13}$$

Equation A-12 becomes:

$$\lambda_k^i(m) = \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(d_k = i, S_k = m, R_k | S_{k-1} = m', R_1^{k-1})\ P(S_{k-1} = m', R_1^{k-1})$$

(A-14)

Events after time $k-1$ do not depend on R_1^{k-1} if the state S_{k-1} is known, then Equation A-14 becomes:

$$\lambda_k^i(m) = \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(d_k = i, S_k = m, R_k | S_{k-1} = m')\ P(S_{k-1} = m' | R_1^{k-1})\ P(R_1^{k-1})$$

(A-15)

In Equation A-15, the term $P(R_1^{k-1})$ will be neglected because it is independent of m or m' and will be cancelled when substituting into Equation A-6.

Equation A-15 then becomes:

$$\lambda_k^i(m) = \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(d_k = i, S_k = m, R_k | S_{k-1} = m')\ P(S_{k-1} = m' | R_1^{k-1}) \qquad \text{(A-16)}$$

Substituting $\lambda_k^i(m)$, i =0, 1 from Equation A-16 to the logarithmic of likelihood ratio (LLR) $\Lambda(d_k)$ in Equation A-6:

$$\Lambda(d_k) = \log \frac{\sum_m \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(S_{k-1} = m' | R_1^{k-1})\ P(d_k = 1, S_k = m, R_k | S_{k-1} = m')}{\sum_m \sum_{m'} P(R_{k+1}^N | S_k = m)\ P(S_{k-1} = m' | R_1^{k-1})\ P(d_k = 0, S_k = m, R_k | S_{k-1} = m')}$$

(A-17)

To calculate $\Lambda(d_k)$ in Equation A-17, as in [87] the probability functions $\alpha_k(m)$, $\beta_k(m)$, and $\gamma_i(R_k, m', m)$ are defined by:

$$\alpha_k(m) = P(S_k = m \mid R_1^k) \tag{A-18a}$$

$$\beta_k(m) = \frac{P(R_{k+1}^N \mid S_k = m)}{P(R_{k+1}^N \mid R_1^k)} \tag{A-18b}$$

$$\gamma_i(R_k, m', m) = P(d_k = i, S_k = m, R_k \mid S_{k-1} = m') \tag{A-18c}$$

Equation A-17 then becomes:

$$\Lambda(d_k) = \log \frac{\sum_m \sum_{m'} \gamma_1(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)} \tag{A-19}$$

The probability $\gamma_i(R_k, m', m)$ in Equation A-18c can be calculated as:

$$\gamma_i(R_k, m', m) = P(d_k = i, S_k = m, R_k \mid S_{k-1} = m')$$

$$= \frac{P(d_k = i, S_k = m, R_k, S_{k-1} = m')}{P(S_{k-1} = m')}$$

$$= \frac{P(R_k \mid d_k = i, S_{k-1} = m', S_k = m)\ P(d_k = i, S_{k-1} = m', S_k = m)}{P(S_{k-1} = m')}$$

$$= \frac{P(R_k \mid d_k = i, S_{k-1} = m', S_k = m)\ P(d_k = i \mid S_{k-1} = m', S_k = m)\ P(S_{k-1} = m', S_k = m)}{P(S_{k-1} = m')}$$

$$= \frac{P(R_k \mid d_k = i, S_{k-1} = m', S_k = m)\ P(d_k = i \mid S_{k-1} = m', S_k = m)\ P(S_k = m \mid S_{k-1} = m')\ P(S_{k-1} = m')}{P(S_{k-1} = m')}$$

$$= P(R_k \mid d_k = i, S_{k-1} = m', S_k = m)\ P(d_k = i \mid S_{k-1} = m', S_k = m)\ P(S_k = m \mid S_{k-1} = m') \quad \text{(A-20)}$$

where $P(R_k \mid d_k = i, S_{k-1} = m', S_k = m)$ is the transition probability of the discrete Gaussian memoryless channel. For rate 1/2 encoder, R_k is made up of two uncorrelated Gaussian random variables x_k, y_k (i.e., $R_k = (x_k, y_k)$).

Then:

$$P(R_k \mid d_k = i, S_{k-1} = m', S_k = m) = P(x_k \mid d_k = i, S_{k-1} = m', S_k = m)\ P(y_k \mid d_k = i, S_{k-1} = m', S_k = m)$$

(A-21)

Since the convolutional encoder is a deterministic machine, $P(d_k = i \mid S_{k-1} = m', S_k = m)$ is equal to 1 if the transition from $S_{k-1} = m'$ to $S_k = m$ due to input $d_k = i$ exists and equal to 0 if it does not exist. The transition state probabilities $P(S_k = m \mid S_{k-1} = m')$ of the trellis are defined by the encoder input statistics. Generally, $P(d_k = 1) = P(d_k = 0) = 1/2$. Since there are two possible transitions from each state, then $P(S_k = m \mid S_{k-1} = m') = 1/2$ for each of these transitions.

The probabilities $\alpha_k(m)$ and $\beta_k(m)$ can be recursively calculated from $\gamma_i(R_k, m', m)$ as in [4]:

$$\alpha_k(m) = \frac{\sum_{m'} \sum_{i=0}^{1} \gamma_i(R_k, m', m)\ \alpha_{k-1}(m')}{\sum_{m} \sum_{m'} \sum_{i=0}^{1} \gamma_i(R_k, m', m)\ \alpha_{k-1}(m')} \quad \text{(A-22)}$$

$$\beta_k(m) = \frac{\sum_{m'}\sum_{i=0}^{1}\gamma_i(R_{k+1}, m', m)\ \beta_{k+1}(m')}{\sum_{m}\sum_{m'}\sum_{i=0}^{1}\gamma_i(R_{k+1}, m', m)\ \alpha_k(m')} \tag{A-23}$$

The MAP algorithm implementation:

1. Initialize the boundary conditions:

   a. Suppose the trellis starts with the zero state, then:

$$\alpha_0(m) = \begin{cases} 1 & m = 0 \\ 0 & m \neq 0 \end{cases} \tag{A-24a}$$

   b. If the trellis is terminated and the last state is the zero state, then:

$$\beta_N(m) = \begin{cases} 1 & m = 0 \\ 0 & m \neq 0 \end{cases} \tag{A-24b}$$

   If the trellis is not terminated (i.e., the final state is known), then $\beta_N(m) = \frac{1}{2^{\nu}}$ for all m at time N.

2. As soon as $R_k = (x_k, y_k)$ is received, the decoder computes $\gamma_i(R_k, m', m)$ and $\alpha_k(m)$ using Equations A-20 and A-22, respectively.

3. At time k, repeat Step 2 for all $m = 0, 1, \ldots, 2^{\nu}-1$, where ν is the memory of the encoder.

4. Repeat Steps 2 and 3 for all time $k = 0, \ldots, N$.

5. The obtained values of $\alpha_k(m)$ and $\gamma_i(R_k, m', m)$ are stored for all k and m.

6. After the complete sequence R_1^N has been received, the decoder recursively computes $\beta_k(m)$ using Equation A-23.

7. At time k, repeat Step 6 for all $m = 0, 1, \ldots, 2^{\nu}-1$, where ν is the memory of the encoder with $k = N-1$.

8. Repeat Steps 6 and 7 for all time $k = N-2, N-3, \ldots, 0$.

9. The obtained values of $\beta_k(m)$ are stored for all k, and m.

10. For $k = 0, 1, \ldots, N-1$, having $\alpha_k(m)$ and $\gamma_i(R_k, m', m)$, the logarithmic of likelihood ratio (LLR), $\Lambda(d_k)$ associated with each decoded bit d_k is computed using Equation A-19.

The MAP algorithm calculates the APP for each decoded bit. However, the MAP algorithm suffers from a computational complexity due to working with a large number of multiplications. The log-MAP algorithm [57] performs the MAP algorithm in the logarithmic domain, i.e., performing multiplications in the logarithmic domain as additions. The log-MAP algorithm is a transformation of MAP, which has equivalent performance without its problems in practical implementations.

References

[1] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Systems Technical Journal,* pp.1-10, January 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: TURBO-CODES," *in Proceeding of 1993 IEEE International Conference On Communication* (Geneva, Switzerland, 1993), pp. 1064-1070.

[3] S. H. Redl, M. K. Weber, and M. W. Oliphant, *An Introduction to GSM*, Artech house, Inc, 1995

[4] R. L. Peterson, R. E. Zeimer, and D. E. Borth, *Introduction to Spread Spectrum Communications*, Prentice-Hall, Inc, 1995.

[5] D. Divsalar and F. Pollara, "Multiple Turbo Codes" *Proceeding of IEEE Military Communications MILCOM'95*, San Diego, California, November 5-8, 1995.

[6] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design and iterative decoding," *IEEE Transactions on Communications*, May 1998, pp. 909-929.

[7] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenated trellis coded modulation with iterative decoding," *IEEE International Symposium on Information Theory*, 1997, pp. 8.

[8] D. Divsalar and F. Pollara, "Serial and Hybrid concatenated codes with applications," *International Symposium on Turbo codes & related topics,* Brest, France, 1997, pp. 80-87.

[9] D. Divsalar, and F. Pollara, "Hybrid concatenated codes and iterative decoding," *The Telecommunications and Data Acquisition Progress (TDA) progress Report 42-130,* Jet Propulsion Lab (JPL), *Pasadena*, CA.

[10] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems-Part I: Characterization," *IEEE Communication Magazine*, pp. 90-100, July 1997.

[11] B. Sklar, *Digital Communications: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1988.

[12] J. G. Proakis, *Digital Communications*, New York: McGraw-Hill, 3rd ed., 1995.

[13] J. D. Parsons, *The Mobile Radio Propagation Channel*, New York, John Wiley&Sons, 1992.

[14] T. S. Rappaport, *Wireless Communications*, Upper Saddle River, NJ: Prentice Hall, 1996.

[15] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communication Systems-Part II: Mitigation," *IEEE Communication Magazine*, pp. 102-109, July 1997.

[16] G. D. Forney, Jr., *Concatenated Codes*, Cambridge, Massachusetts: Massachusetts Institute of Technology, 1966.

[17] A. Mehrotra, *GSM System Engineering*, Artech House, Inc., 1997

[18] S. Lin and D. J. Costello, *Error Control Coding: fundamentals and applications*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[19] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, April 1950.

[20] P. A. Ramsdale, "Personal Communications in the UK-Implementation of PCN using DCS 1800," *in International Journal on Wireless information Networks*, Vol. 1, pp. 29-36, 1994.

[21] J. P. Odenwalder, *Error Control Coding Handbook*, Linkabit Corporation, San Diego, California, 1976.

[22] S. G. Wilson, *Digital Modulation and Coding*, Upper Saddle River, NJ: Prentice-Hall, 1996.

[23] J. L. Massey, "The How and Why of channel coding," *Proceeding of 1984 Zurich Seminar on Digital Communications*, pp. 67-73, 1984.

[24] S. B. Wicker and V. K. Bharagava, *Reed-Solomon Codes and their Applications*, IEEE Press, New York, 1994.

[25] G. Battail, C. Berrou, and A. Glavieux, "Pseudo-random recursive convolutional coding for near-capacity performance," *in Proceeding of Communication Theory Mini-Conference., GLOBECOM'93* (Houston, TX, Dec. 1993), pp. 23-27

[26] S. Benedetto and G. Montorsi, "Role of recursive convolutional codes in turbo codes," *Electronic. Letters*, Vol. 31, No. 11, pp. 858-859, May 25, 1995.

[27] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-codes," *IEEE Transactions on Commmunications*, Vol. 44, No. 10, pp. 1261-1271, Oct. 1996.

[28] D. Divsalar and F. Pollara, "On the design of turbo codes," *The Telecommunications and Data Acquisition Progress (TDA) progress Report 42-123,* Jet Propulsion Lab (JPL), *Pasadena*, CA, pp.99-121, Nov. 15, 1995.

[29] D. Divsalar and R. J. McEliece, "Effective free distance of turbo codes," *Electronic Letters, Vol. 32, no. 5, Feb. 1996.*

[30] M. S. C. Ho, S. S. Pietrobon, and T. Giles, "Optimising the constitute codes for turbo decoders," *International Symposium on Information Theory ISIT 1998, Cambridge, MA, USA*, Aug. 1998.

[31] S. Benedetto, R. Garello, and G. Montorsi, "A search for good convolutional codes to be used in the construction of turbo codes," *IEEE Transactions on Communications Vol. 46, no. 9, Sep. 1998, pp.* 1101-1105.

[32] P. Robertson, "Illuminating the structure of parallel concatenated recursive systematic (TURBO) codes," *in Proceeding of Global Communications GLOBECOM'94* (San Francisco, CA, Nov. 1994), Vol. 3, pp. 1298-1303.

[33] A. S. Barbulescu and S. S. Pietrobon, "Terminating the trellis of turbo-codes in the same state," *Electronic. Letters,* Vol. 31, No. 1, pp. 22-23, Jan. 5, 1995.

[34] W. J. Blackert, E. K. Hall and S. G. Wilson, "Turbo code termination and interleaver conditions," *Electronic Letters*, Vol. 31, No. 24, pp. 2082-2084, Nov. 23, 1995.

[35] P. Robertson, "Improving decoder and code structure of parallel concatenated recursive systematic (turbo) codes," *in Proceeding of IEEE International Conference on Universal Personal Communication ICUPC'94*, San Diego, 1994,pp. 183-187.

[36] P. Jung and M. Nabhan, "Performance evaluation of turbo-codes for short frame transmission systems," *Electronic Letters,* Vol. 30, No. 2, pp. 111-113, Jan. 20, 1994.

[37] M. C. Reed and S. S. Pietrbon, "Turbo code termination schemes and a novel alternative for short frames," *proceeding of Personal, Indoor and Mobile Radio Communication (PIMRC'96), Taipai, Taiwan, 1996.*

[38] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," *in Proceeding of 1995 IEEE International Conference On Communications* (Seattle, WA, June 1995).

[39] D. Divsalar and F. Pollara, "Turbo codes for deep-space communications," *The Telecommunications and Data Acquisition Progress (TDA) progress Report 42-120,* Jet Propulsion Lab (JPL), *Pasadena*, CA, pp.29-39, Feb. 15, 1995.

[40] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," *The Telecommunications and Data Acquisition Progress (TDA) progress Report 42-121,* Jet Propulsion Lab (JPL), *Pasadena*, CA, pp.66-77, May 15, 1995.

[41] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *The Telecommunications and Data Acquisition Progress (TDA) progress Report 42-122,* Jet Propulsion Lab (JPL), *Pasadena*, CA, pp.56-65, Aug. 15, 1995.

[42] S. Benedetto and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, Vol. 44, No. 5, pp. 591-600, May 1996.

[43] P. Komulainen and K. Pehkonen, "Performance Evaluation of Superorthogonal Turbo Codes in AWGN and Flat Rayleigh Fading Channels," *IEEE Journal on Selected Area in Communications,* Vol. 16, No. 2, Feb. 1998, pp. 196-205.

[44] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-output decoding algorithms in iterative decoding of parallel concatenated convolutional codes," *in Proceeding of 1996 IEEE International Conference On Communications* (Dallas, Texas, June 1995), pp. 112-117.

[45] B. Vucetic and Y. Li, "A Survey of soft-output Algorithms," *Proceeding of International Symposium on Information Theory and its Applications ISITA'94*, Sydney, Australia, pp. 863-867 November 1994.

[46] G. D. Forney, "The Viterbi algorithm," *Proceeding of IEEE*, Vol. 61, No. 3, pp. 268-278, Mar. 1973.

[47] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, pp. 284-287, March 1974.

[48] K. Abend and B. D. Fritchman, "Statistical Detection for Communication on Channel With Intersymbol Interference," *Proceeding of IEEE*, Vol. 58, No. 5, pp. 779-785, May 1970.

[49] B. Sklar, "A primer on Turbo Code Concepts," *IEEE Communication Magazine*, pp. 94-101, Dec. 1997.

[50] C. Berrou, P. Adde, E. Angui and S. Faudeil, "A low Complexity Soft-output Viterbi Decoder Architecture," *Proceeding of Proceeding of 1995 IEEE International Conference On Communications ICC'93, Geneva*, pp. 737-740, May 1993.

[51] J. Hagenauer and P. Hoeher, "A Vitrbi Algorithm with Soft-Decision Outputs and its Applications," *Proc. of Global Communications Globecom'89*, Dallas, Vol. 3, pp. 47.1.1-47.1.7, Nov. 1989.

[52] L. Pake, P. Robertson, and E. Villebrum, "Improved Decoding with the SOVA in a Parallel Concatenated (Turbo-Code) Scheme," *in Proceeding of 1996 IEEE International Conference On Communications ICC'96*, Dallas, TX, pp. 102-106, June 1996.

[53] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," *in Proceeding of 1995 IEEE International Conference On Communications ICC'95*, Seattle, Washington, pp. 1009-1013, June 1995.

[54] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes," *The Telecommunications and Data Acquisition Progress Rep. 42-127*, November 1996, Jet Propulsion Lab. Pasadena, California.

[55] S. S. Pietrobon and A. S. Barbulescu,"A simplification of the Modified Bahl algorithm for Systematic Convolutional codes," *Proceeding of International Symposium on Information Theory and its Applications ISITA'94*, Sydney, Australia, pp. 1073-1077, November 1994.

[56] J. Hagenauer, E. Offer, and L. Papke, "Decoding 'Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, Vol. 42, No. 2, pp. 429-445, Mar. 1996.

[57] P. Robertson, P. Hoeher, and E. Villebrun," Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding," *European Transactions On Telecommunications*, Vol. 8, pp. 119-25, Mar/Apr 1997.

[58] S. Benedetto and G. Montorsi, "Average performance of parallel concatenated block codes," *Electronic Letters,* Vol. 31, No.3, pp. 156-158, Feb. 2, 1995.

[59] S. Benedetto and G. Montorsi, "Performance evaluation of TURBO-codes," *Electronic Lett*ers, Vol. 31, No.3, pp. 163-165, Feb. 2, 1995.

[60] J. Hagenauer and L. Papke, "Decoding 'Turbo'-codes with the soft output Viterbi algorithm," *in Proceeding 1994 IEEE International Symposium on Information Theory* (Trondheim, Norway, June 1994), p. 164.

[61] S. Benedetto and G. Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," *IEEE Transactions on Information Theory*, Vol. 42, No. 2, pp. 409-428, Mar. 1996.

[62] D. Divsalar, S. Dolinar, F. Pollara, and R. J. McEliece, "Transfer function bounds on the performance of turbo codes," *TDA Progr. Rep. 42-122, Jet Propulsion Lab., Pasadena, CA*, pp. 44-55, Aug. 15, 1995.

[63] A. J. Viterbi and J. K. Omura, *Principles of Digital communications and coding.* New York: MacGraw-Hill, 1979.

[64] L.C. Perez, J. Seghers, and D. J. Costello, "A Distance Spectrum Interpretation of Turbo Codes," *IEEE Transactions on Information Theory*, Vol. 42, No.6, pp. 1698-1709, Nov. 1996.

[65] T. M. Duman and M. Salehi, "New performance bounds for turbo codes," *IEEE Transactions on Communications*, June 1998, pp. 717-723.

[66] P. Jung, "Comparison of Turbo-Code Decoders Applied to Short Frame Transmission Systems," *IEEE Journal on Selected Area in Communications*, Vol. 14, No. 3, pp. 530-537, April 1996.

[67] P. Jung and M. Nabhan, "Applying turbo-codes to the up link in a JD-CDMA mobile radio system using coherent receiver antenna diversity," *in proc. ITG Conf.*, Vol. 130, Munich, 1994,pp.49-56.

[68] D. Cygan, F.David, H. J. Eul, J. Hofmann, N. Metzner, and W. Mohr, "RACE-II advanced TDMA mobile access project-an approach for UMTS," *in proc. 1994 Int. Zurich Seminar Digital Communications*, Zurich, Switzerland, 1994, pp. 428-439.

[69] A. S. Barbulescu and S. S. Pietrobon, "Interleaver design for turbo-codes," *Electronic Letters,* Vol. 30, No. 25, pp. 2107, Dec. 8, 1994.

[70] P. Jung and M. Nabhan, "Dependence of the error Performance of turbo-codes on the interleaver structure in short frame transmission systems," *Electronic. Letters,* Vol. 30, No. 4, pp. 285-288, Feb. 17, 1994

[71] H. Gerzberg, "Multilevel Turbo Coding with Short Interleavers," *IEEE Journal on Selected Area in Communications*, Vol. 16, No. 2, pp. 303-309, Feb. 1998.

[72] P. Jung and M. Nabhan, "Application of turbo-codes to a CDMA mobile radio system using joint detection and antenna diversity," *Proceeding of IEEE 44th Vehicular Technology Conference VTC'94*, Stockholm, 1994, pp.770-774.

[73] S. Le. Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation," *in Proceeding of 1994 IEEE International Conference On Communications ICC'94*, May 1994, pp. 645-649.

[74] P. Jung, "Novel low complexity decoder for turbo-codes," *Electronic Letters,* Vol. 31, No. 2, pp. 86-87, Jan. 19, 1995.

[75] Eric K. Hall and Stephen G. Wilson, "Design and Analysis of Turbo Codes on Rayleigh Fading Channels," *IEEE Journal on Selected Area in Communications*, Vol. 16, No. 2, pp. 160-174, Feb. 1998.

[76] B. E. Wahlen and C. Y. Mai, "Turbo coding applied tp pragmatic trellis-coded modulation," *IEEE Communication Letters*, vol. 4, No. 2, pp. 65-67, Feb 2000.

[77] T. M. Duman and M. Salehi, "Performance bounds for turbo coded modulation systems," *IEEE Transactions on Communications*. April 1999, pp. 511-521.

[78] T. M. Duman and M. Salehi, "The union bound for turbo coded modulation systems over fading channels," *IEEE Transactions on Communications*. October 1999, pp. 1495-1502 (see also the correction in the November 1999 issue of the same transactions, page 1766).

[79] J. Hagenauer, "Rate Compbatible Punctured Convolutional Codes and their applications," *IEEE Transactions on Communications*. April 1988, pp. 389-400.

[80] A. S. Barbulescu and S. S. Pietrobon, "Rate Compatible Turbo Codes," *Electronic Letters*, Vol. 31, pp. 535-536, Mar. 1995.

[81] K. R. Narayanan and G. L. Stuber, "A novel ARQ technique using the turbo coding principle," *IEEE Communications Letters*, pp. 49-51, Mar. 1997.

[82] D. N. Rowitch and L.B. Milstein, "Rate Compatible Punctured Turbo (RCPT) codes in a hybrid FEC/ARQ systems," *IEEE Global Communications GLOBCOM'97*, pp. 55-59.

[83] J. Li. and H. Imai, "Performance of hybrid-ARQ protocols with rate compatible turbo codes," *International Symposium On Turbo codes & related topics*, Brest, France, 1997, pp. 188-191.

[84] P. Jung, J. Plechinger, M. Doetsch and F. M. Berens, "Advances on the application of turbo-codes to data services in third generation mobile networks," *International Symposium On Turbo codes & related topics,* Brest, France, 1997, pp. 135-142.

[85] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, pp. 260-269, April 1967.

[86] J. K. Omura, "On the Viterbi decoding algorithms," *IEEE Transactions on Information Theory*, pp. 177-179, January 1969.

[87] J. Hagenauer, P. Robertson, and L. Papke, "Iterative (turbo) decoding of systematic convolutional codes with the MAP and SOVA algorithms," *in proc. ITG conference*, pp. 21-29, September 1994.

[88] J. Hagenauer, "Source-Controlled channel decoding," *IEEE Transactions on Communications*, pp. 2449-2457, September 1995.

[89] R. P. Stanley, *Enumerative Combinatorics*, Montery, California: Wadsworth & Brooks/Cole, 1986.

[90] S. Benedetto, E. Biglieri, and V. Castellani, *Digital Transmission Theory.* Englewood Cliffs, NJ, Prentice-Hall, 1987.

[91] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering,* New York: Wiley, 1965.

[92] I. M. Jacobs, "Sequential decoding for efficient communication from deep space," *IEEE Tranactions on Communication Technology*, Vol. Com-15, pp. 492-501, Aug. 1967.

[93] Y. Wu and B. D. Woerner, "The influence of quantization and fixed point arithmetic upon the BER performance of turbo codes," *Proceeding of IEEE Vehicular Technology Conference*, (Huston, TX), May 1999.

[94] W. C. Jakes, *Mobile Microwave Communication,* New York: John Wiley & sons 1974.

[95] R. Jain, *The Art of Computer Systems Analysis performance,* John Wiley & Sons 1991.

[96] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*, New York: Plennum Press, 1992.

[97] J. Hokfelt and T. Maseng, "Optimizing the energy of Different Bitstreams of Turbo Codes," *Turbo Coding Seminar Proceedings, Lund, Sweden*, pp.59-63, Aug. 1996.

[98] T. M. Duman and M. Salehi, "On Optimal Power Allocation for Turbo Codes," in *Proceeding of IEEE International Symposium On Inforamation Theory,* Ulm., Germany,1997, pp.104.

[99] M. Oberg and P.H. Siegel, "The effect of puncturing in turbo encoders," *International Symposium On Turbo codes & related topics*, Brest, France, pp.184-187, Septemper 1997.

[100] M. C. Valenti and B. D. Woerner, "Variable latency turbo codes for wireless multimedia applications," *International Symposium On Turbo codes & related topics*, Brest, France, pp.216-219, September 1997.

[101] A. S. Barbulescu, *Iterative decoding of turbo codes and other concatenated codes*, Ph. D. thesis, University of South Australia, February 1996.

[102] S. Benedetto and G. Montorsi, "Design guidelines of parallel concatenated convolutional codes," *IEEE Global Communications GLOBECOM*, pp. 2273-2277, November 1995.

[103] D. Divsalar, S. Dolinar, R. J. McEliece, and F. Pollara, "Performance analysis of turbo codes," *IEEE Military Communications MILCOM*, pp. 91-96, November 1995.

[104] J. B. Cain, G. C. Clark, and J. M. Geist, " Punctured convolutional codes of rate (n-1)/n and simplifyied maximum likelihood decoding," *IEEE Transactions on Information Theory*, Vol. IT-25, pp. 97-100, January 1979.

[105] D. Haccoun and G. Begin, "High rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Transactions on Communications*, Vol. 37, pp. 1113-1125, November 1989.

[106] G. Begin, D. Haccoun, and C. Paquin, " Further results on high rate punctured convolutional codes for Viterbi and sequential decoding," *IEEE Transactions on Communications*, Vol. 38, pp. 1922-1928, November 1990.

[107] J. Hagenauer, N. Seshadri, and C.W. Sundberg, "The performance of rate compatible punctured convolutional codes for digital mobile radio," *IEEE Transactions on Communications*, Vol. 38, pp. 966-980, July 1990.

[108] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics,* Englewood Cliffs, NJ, Prentice Hall, 1995, pp. 517.

[109] F. Gagnon and D. Haccoun, "Bounds of the error performance of coding for nonindependent Rician-fading channels," *IEEE Transactions on Communications,* Vol. 40, pp. 351-360, February 1992.

[110] S. Shamai and G. Kaplan, "Achievable performance over the correlated Rician channel," *IEEE Transactions on Communications,* Vol.42, pp. 2967-2978, November 1994.

[111] M. Valenti and B. Woerner, "Performance of turbo codes in interleaved flat fading channels with estimated channel state information," *in Proceeding IEEE Vehicular Technology Conference,* (Ottawa Canada), pp. 66-70, May 1998.

[112] M. M. Salah, R. A. Raines, M. A. Temple, and T. G. Bailey, "Energy allocation strategies for turbo codes with short frames," *IEEE International Conference on Information Technology: Coding and Computing,* Las Vegas, Nevada, pp.408-411, March 2000.

[113] M. M. Salah, R. A. Raines, M. A. Temple, and T. G. Bailey, "A general interleaver for equal and unequal error protections of turbo codes with short frames," *IEEE International Conference on Information Technology: Coding and Computing,* Las Vegas, Nevada, pp.412-415, March 2000.

[114] M. M. Salah, R. A. Raines, M. A. Temple, and T. G. Bailey, "Approach for deriving performance bounds of punctured turbo codes," *IEE Electronic Letters*, Vol. 35, No. 25, pp. 2191-2192, December 1999.

[115] A. J. Viterbi, "Wireless Digital Communication: A view based on three lesson learned," *IEEE Communication Magazine, September 1991,* pp. 33-36.

Vita

Major Moataz Salah was born in Cairo, Egypt on October 20, 1966. In 1988, he received his Bachelor of Science degree in Electrical Engineering, receiving a general grade of "Excellent" from the Military Technical College (MTC), Cairo, Egypt. In 1989, he was appointed an Assistance Instructor in MTC with responsibility for teaching Communication Theory, Digital Signal Processing, Communication Networks and related subject matter. In 1994, he received his Master of Science degree in Electrical Engineering from MTC. In February 1994, he was promoted to Assistant Lecturer in MTC, retaining primary faculty responsibilities in communications and signal processing. In September 1996, Major Salah was competitively selected by the Egyptian government granted and a fellowship to the Graduate School of Engineering and Management, Air Force Institute of Technology to pursue a doctorate degree in Electrical Engineering. Major Moataz Salah is a member of IEEE Communications and Information Societies and two honor societies, Tau Beta Pi and Eta Kappa Nu.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 2000	PhD Dissertation

4. TITLE AND SUBTITLE

Turbo Codes for Wireless Mobile Communication Systems Applications

5. FUNDING NUMBERS

6. AUTHOR(S)

Moataz M. Salah, Major, Egypt

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENG/00-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Major Richard A. Raines, ENG, DSN: 785-3636 ext 4715

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

ABSTRACT (*Maximum 200 Words*)

In this dissertation, different ways to enhance the performance of turbo codes with short frames are presented. One way of enhancing the performance of turbo codes with short frames is by optimizing the energy allocation strategies to the output bitstreams. For turbo codes with short frames, different ways to allocate the energy are investigated using computer simulation and modified analytic bounds. The results show that the performance is improved without any increase in complexity. Another way to enhance the performance is with the proper design of the interleaver. This work proposes a new and unique interleaver for equal and unequal error protections.

A novel analytical bound is developed to evaluate the performance of punctured turbo codes, along with its applications to different channel types. The new approach introduces a random device, the *hypergeometric-puncturing device*, which averages the output weight of the punctured codeword over all the punctured positions. This bound serves to illustrate the achievable performance of turbo codes and the effects of block length and constituent encoder choice in the performance of turbo codes.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Channel coding techniques, error control codes, turbo codes, wireless communications, short frame applications	218
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102